

T S1/5/1

1/5/1

DIALOG(R)File 351:Derwent WPI

(c) 2005 Thomson Derwent. All rts. reserv.

012079124 **Image available**

WPI Acc No: 1998-496035/199842

XRPX Acc No: N98-387398

Scope testing method for documents in hierarchical file system - involves creating cache of parent directory identities and using backwards file system pointers to check if scope is correct

Patent Assignee: MICROSOFT CORP (MICR-N)

Inventor: MILEWSKI B B; PELTONEN K G; RAJU S C V

Number of Countries: 019 Number of Patents: 004

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 9839700	A2	19980911	WO 98US4568	A	19980306	199842 B
US 5890147	A	19990330	US 97813618	A	19970307	199920
EP 920666	A2	19990609	EP 98910257	A	19980306	199927
			WO 98US4568	A	19980306	
JP 2000510986	W	20000822	JP 98538909	A	19980306	200045
			WO 98US4568	A	19980306	

Priority Applications (No Type Date): US 97813618 A 19970307

Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 9839700	A2	E	49	G06F-000/00	
				Designated States (National): DE GB JP	
				Designated States (Regional): AT BE CH DE DK ES FI FR GB GR IE IT LU MC NL PT SE	
US 5890147	A			G06F-017/30	
EP 920666	A2	E		G06F-017/30	Based on patent WO 9839700
				Designated States (Regional): DE FR GB	
JP 2000510986	W		51	G06F-017/30	Based on patent WO 9839700

Abstract (Basic): WO 9839700 A

The search system allows documents to be identified from a set that includes their location in a hierarchical filing system. The search system (64) is run on a computer with access to a set of file structures (26-30). Each disc in the structure has a file hierarchy and a mapping (80-84) between each file and its parent files. Each disc also has an index (72-76) that is used to identify documents having a defined criteria.

As documents matching the requested criteria are returned, the identity of their parent is found and entered in a cache (86) along with a 'scope' flag. Subsequent retrieved files then check the cache to determine if they are in scope.

ADVANTAGE - By dynamically building an in-scope parent table, scope testing resources are reduced.

Dwg.3/15

Title Terms: SCOPE; TEST; METHOD; DOCUMENT; HIERARCHY; FILE; SYSTEM; CACHE; PARENT; DIRECTORY; IDENTIFY; BACKWARD; FILE; SYSTEM; POINT; CHECK; SCOPE; CORRECT

Derwent Class: T01

International Patent Class (Main): G06F-000/00; G06F-017/30

International Patent Class (Additional): G06F-012/00

File Segment: EPI

?

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号
特表2000-510986
(P2000-510986A)

(43) 公表日 平成12年8月22日 (2000.8.22)

(51) Int.Cl. ⁷	識別記号	F I	テマコード* (参考)
G 0 6 F 17/30		G 0 6 F 15/40	3 5 0 B
12/00	5 2 0	12/00	5 2 0 E
			5 2 0 P
		15/40	3 1 0 F
		15/411	3 1 0
審査請求 未請求 予備審査請求 未請求 (全 51 頁)			

(21) 出願番号 特願平10-538909
(86) (22) 出願日 平成10年3月6日 (1998.3.6)
(85) 翻訳文提出日 平成10年11月9日 (1998.11.9)
(86) 国際出願番号 PCT/US98/04568
(87) 国際公開番号 WO98/39700
(87) 国際公開日 平成10年9月11日 (1998.9.11)
(31) 優先権主張番号 08/813, 618
(32) 優先日 平成9年3月7日 (1997.3.7)
(33) 優先権主張国 米国 (US)
(81) 指定国 EP (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, L U, MC, NL, PT, SE), DE, GB, J P

(71) 出願人 マイクロソフト コーポレイション
アメリカ合衆国 98052 ワシントン州
レッドモンド ワン マイクロソフト ウ
エイ (番地なし)
(72) 発明者 ベルトネン, カイル, ジー.
アメリカ合衆国 98027 ワシントン州
アイザッカー サウスイースト 42エヌデ
イー プレイス 18126
(72) 発明者 ラジュ, シタラム, シー., ヴィー.
アメリカ合衆国 98029 ワシントン州
アイザッカー 245ティーエイチ アヴェ
ニュー サウスイースト 4325
(74) 代理人 弁理士 谷 義一 (外2名)

最終頁に続く

(54) 【発明の名称】 サーチエンジンにおけるドキュメントのスコープ・テスト

(57) 【要約】

ドキュメントおよびフォルダの階層編成システムにおいて照会に回答する方法およびメカニズムが開示されている。照会に回答して、ドキュメントのセットは指定された基準に基づいて検索される。そのセット内にあって、指定されたスコープに合致するドキュメントだけが結果セットに入って戻される。スコープ・テストはセット内のドキュメントの各々について行われ、これは各ドキュメントのドキュメント識別子を取得し、そのドキュメント識別子を使用してその親フォルダのドキュメント識別子を取得することによって行われる。親フォルダのドキュメント識別子はデータ構造へのキーとして使用され、データ構造は親フォルダが指定されたスコープ内にあるかどうかを示すフラグをストアしている。ある親フォルダのフラグが親フォルダがスコープ内にあることを示していれば、その親をもつドキュメントは結果セットに入って戻される。フラグがカレントドキュメントがスコープ内にないことを示していれば、そのドキュメントは戻されない。そのキーに対応するエントリがデータ構造になかったときは、プレフィックス・マッチングが親フォ

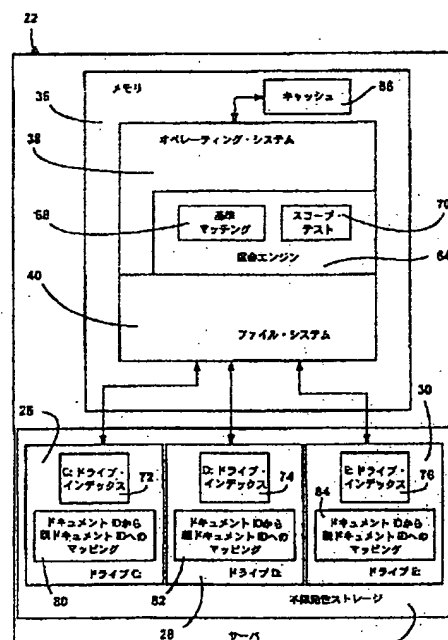


FIG. 3

【特許請求の範囲】

1. ドキュメントおよびフォルダの階層編成システムにおいて、指定された基準に基づいて検索された1つまたは2つ以上のドキュメントの中から、指定されたスコープに合致する少なくとも1つのドキュメントでもって照会に応答する方法であって、該方法は、

検索されたドキュメントをカレントドキュメントとして選択するステップと、

前記カレントドキュメントの識別子を取得するステップと、

前記カレントドキュメントの親フォルダの識別子を取得するステップと、

データ構造にアクセスするステップであって、該データ構造は前記親フォルダの前記識別子でインデックスされたフラグを含み、該フラグは前記カレントドキュメントがスコープ内にあるかどうかを示す値をもつステップと、

前記フラグの前記値があらかじめ決められた値に等しい場合には、前記照会に
応答して前記カレントドキュメントの情報を戻すステップと

を備えることを特徴とする方法。

2. 請求項1に記載の方法において、親フォルダの識別子を取得する前記ステップは、ドキュメント識別子から親ドキュメント識別子へのマッピング・テーブルから前記親フォルダの前記識別子を検索するステップを備えることを特徴とする方法。

3. 請求項2に記載の方法において、前記親フォルダの前記識別子を取得する前記ステップは、前記カレントドキュメントの前記ドキュメント識別子を、前記ドキュメント識別子から親ドキュメント識別子へのマッピング・テーブルへのインデックスとして使用するステップを含むことを特徴とする方法。

4. 請求項1に記載の方法において、前記カレントドキュメントの情報を戻す前記ステップは、前記カレントドキュメントに関する情報を結果セットに入れるステップを備えることを特徴とする方法。

5. 請求項1に記載の方法において、前記データ構造はハッシュ・テーブルを備えることを特徴とする方法。

6. 請求項1に記載の方法において、別の検索されたドキュメントを第2カレ

ントドキュメントとして選択するステップと、前記第2カレントドキュメントの識別子を取得するステップと、前記第2カレントドキュメントの第2親フォルダの識別子を取得するステップと、前記第2親フォルダの前記識別子をインデックスとして前記データ構造にアクセスするステップと、前記第2親フォルダの前記識別子が前記データ構造内のエントリに一致しないことを示す値を前記データ構造から受け取るステップと、前記第2親フォルダを前記指定されたスコープと比較するステップと、前記第2親フォルダの前記識別子でインデックスされたエントリを前記データ構造に追加するステップとをさらに備え、前記エントリは前記第2親フォルダが前記指定されたスコープに一致するかどうかを示す値を含むことを特徴とする方法。

7. 請求項6に記載の方法において、前記第2親フォルダを前記指定されたスコープと比較する前記ステップは、前記第2親フォルダの階層的に上位にある第3フォルダの識別子を取得するステップと、第3親フォルダの前記識別子をインデックスとして前記データ構造にアクセスするステップとを含むことを特徴とする方法。

8. 請求項6に記載の方法において、前記第2親フォルダが前記指定されたスコープに一致する場合には、前記第2親フォルダの前記識別子でインデックスされた入力値はあらかじめ決められた値に等しいことを特徴とする方法。

9. 請求項1に記載の方法において、前記親フォルダは前記カレントドキュメントの直属親フォルダであることを特徴とする方法。

10. 請求項1に記載の方法において、前記照会は少なくとも1つのフォルダを指定しており、前記方法は、前記少なくとも1つのフォルダの識別子でインデックスされたフラグを前記データ構造に初期値として入れておくステップをさらに備え、前記フラグは前記少なくとも1つのフォルダがスコープ内にあることを示す値をもつことを特徴とする方法。

11. 請求項1に記載の方法において、前記フラグは前記ドキュメントがスコープ内にあることを示し、前記方法は、前記親フォルダの階層的に下位にあるフォルダの識別子を取得するステップと、前記親フォルダの下位にある前記フォル

ダの前記識別子でインデックスされた少なくとも1つの追加フラグを前記データ構造に入れるステップとをさらに備え、前記フラグは、前記親フォルダの下位にある前記フォルダがスコープ内にあることを示す値をもつことを特徴とする方法。

12. 請求項1に記載の方法において、前記階層編成システムはファイル・システムであることを特徴とする方法。

13. ドキュメントおよびフォルダの階層編成システムにおいて、指定された基準に基づいて検索された1つまたは2つ以上のドキュメントの中から、指定されたスコープに合致する少なくとも1つのドキュメントでもって照会に応答するメカニズムであって、

検索されたドキュメントをカレントドキュメントとして選択する手段と、

前記カレントドキュメントの識別子を取得する手段と、

前記カレントドキュメントの親フォルダの識別子を取得する手段と、

データ構造にアクセスする手段であって、該データ構造は前記親フォルダの前記識別子でインデックスされたフラグを含み、該フラグは前記カレントドキュメントがスコープ内にあるかどうかを示す値をもつ手段と、

前記フラグの前記値が前記ドキュメントがスコープ内にあることを示す場合には、前記照会に응答して前記カレントドキュメントの情報を戻す手段と

を備えることを特徴とするメカニズム。

14. 請求項13に記載のメカニズムにおいて、親フォルダの識別子を取得する前記手段は、ドキュメント識別子から親ドキュメント識別子へのマッピング・テーブルを含むことを特徴とするメカニズム。

15. 請求項13に記載のメカニズムにおいて、前記データ構造はハッシュ・テーブルを備えることを特徴とするメカニズム。

16. 請求項13に記載のメカニズムにおいて、前記階層編成システムはファイル・システムであることを特徴とするメカニズム。

17. 請求項13に記載のメカニズムにおいて、別の検索されたドキュメントを第2カレントドキュメントとして選択する手段と、前記第2カレントドキュメ

ントの識別子を取得する手段と、前記第2カレントドキュメントの第2親フォルダの識別子を取得する手段と、前記第2親フォルダの識別子をインデックスとしてデータ構造にアクセスする手段と、前記第2親フォルダの前記識別子が前記データ構造内のエントリに一致しないことを示す値を前記データ構造から受け取る手段と、前記第2親フォルダを前記指定されたスコープと比較する手段と、前記第2親フォルダの前記識別子でインデックスされたエントリを前記データ構造に追加する手段とをさらに備え、前記エントリは前記第2親フォルダが前記指定されたスコープに一致するかどうかを示す値を含むことを特徴とするメカニズム。

18. 請求項17に記載のメカニズムにおいて、前記第2親フォルダを前記指定されたスコープと比較する前記手段は、前記第2親フォルダの階層的に上位にある第3フォルダの識別子を取得する手段と、第3親フォルダの前記識別子をインデックスとして前記データ構造にアクセスする手段とを含むことを特徴とするメカニズム。

19. 請求項13に記載のメカニズムにおいて、前記カレントドキュメントの前記親フォルダは直属親フォルダであることを特徴とするメカニズム。

20. 請求項13に記載のメカニズムにおいて、前記フラグは前記ドキュメントがスコープ内でないことを示し、前記メカニズムは、前記親フォルダの階層的に下位にあるフォルダの識別子を取得する手段と、前記親フォルダの下位にある前記フォルダの前記識別子でインデックスされた少なくとも1つの追加フラグを前記データ構造に入れる手段とをさらに備え、前記フラグは、前記親フォルダの下位にある前記フォルダがスコープ内でないことを示す値をもつことを特徴とするメカニズム。

21. 照会エンジンにおいて、照会に応答するメカニズムであって、
ユーザ指定の制限に応答して1つまたは2つ以上のドキュメントを検索する基準マッチング・コンポーネントと、

ユーザ指定のスコープに応答して検索されたドキュメントの各々を選択するスコープ・テスト・コンポーネントであって、前記ドキュメントのうち選択された1つのドキュメントの識別子を取得する手段と、カレントドキュメントの前記識

別子でインデックスされたテーブルにアクセスして前記カレントドキュメントの上位にある親フォルダの識別子を取得する手段と、前記親フォルダの前記識別子でインデックスされたデータ構造にアクセスする手段であって、該データ構造は前記親フォルダの下位にストアされたドキュメントのスコープを示すフラグを含む手段と、前記フラグがあらかじめ決められた値に等しい場合には、前記照会に応答して前記カレントドキュメントの情報を戻す手段とを含むコンポーネントとを備えることを特徴とするメカニズム。

22. ユーザ指定の制限に応答してドキュメントを検索する基準マッチング・コンポーネントをもつ照会エンジンにおいて、スコープ指定に応答して複数のドキュメントのうちの少なくとも1つのドキュメントのスコープを判断する方法であって、該方法は、

検索されたドキュメントをカレントドキュメントとして選択し、前記カレント

ドキュメントの識別子を取得し、前記カレントドキュメントの親フォルダの識別子を取得し、前記親フォルダの前記識別子をインデックスとして使用してデータ構造にアクセスするステップであって、該データ構造は親フォルダの識別子でインデックスされたフラグを含むステップと、

前記データ構造から応答を受け取るステップであって、該応答は前記カレントドキュメントがスコープ内にあるかどうかを示す値をもつフラグ、または前記親フォルダが前記データ構造内でインデックスされていなかったとの通知を備えるステップと、

前記フラグを受け取ったとき、その値が前記ドキュメントがスコープ内にあることを示すかどうかを判断し、スコープ内にあれば、前記照会に応答して前記カレントドキュメントの情報を戻すステップと、

前記親フォルダがインデックスされていなかったとの通知を受け取ったとき、前記親フォルダを前記スコープ指定と比較して前記カレントドキュメントのスコープを判断し、前記親フォルダでインデックスされたフラグを前記データ構造にストアし、該フラグは前記カレントドキュメントがスコープ内にあるかどうかを示す値をもち、スコープ内にあれば、前記照会に応答して前記カレントドキュメ

ントの情報を戻すステップと
を備えることを特徴とする方法。

23. 請求項22に記載の方法において、前記カレントドキュメントの親フォルダの識別子を取得するステップは、ドキュメントから親へのマッピング・テーブルにアクセスするステップを含むことを特徴とする方法。

【発明の詳細な説明】

サーチエンジンにおけるドキュメントのスコープ・テスト

発明の分野

本発明は一般的にはコンピュータ・サーチエンジン（検索エンジンともいう）に関し、より具体的には、ユーザの照会(query)に応答して、サーチエンジンを使用し、ドキュメント情報を検索する改良方法およびメカニズムに関する。

発明の背景

Microsoft社のオブジェクト・ファイル・システム(OFS:Object File System)やMicrosoft社のWindows NTTMファイル・システム(NTFS:NT File System)といった、先進的コンピュータ・ファイル・システムにおける階層編成データは、サーチエンジン（照会エンジン）と関連づけられている場合がある。この照会エンジンを使用すると、ユーザはデータおよび／またはファイル・システムに問い合わせ（照会）を行い、ユーザの照会で指定したものに合致するドキュメント（つまり、ファイルまたはオブジェクト）を見つけることができる。例えば、上記ファイル・システムはこのシステムによってストアされたドキュメントのコンテンツ（内容）やプロパティのインデックスを別々に作成し、ドキュメント内のデータがデータベース・データのような構造になっていなくても、照会エンジンがそのような照会に即時に応答できるようにしている。照会に応答するために、照会エンジンはインデックスにアクセスし、見つけたドキュメントに関する情報を結果セットに入れて戻している。他の階層編成データを取り扱うサーチエンジンは他にもあるが、その扱い方は類似している。

照会で指定するものの1つに制限(restriction)があるが、これはマッチング（合致する）ドキュメントが所有する基準（コンテンツおよび／またはプロパティ）の集まりである。また、照会で指定するものの代表的なものとしては、調べようとするフォルダまたはディレクトリの集まりであるスコープ(scope:有効範囲)と、結果セットに入って戻される各マッチング・ドキュメントごとにどのプロパ

ティを戻すべきかを特定している戻りセット(return set)がある。例えば、照

会はマッチング・ドキュメントが「コンピュータ・ソフトウェア (computer software)」というテキストを含んでいること、調べるべきスコープがc:\folder]であること、およびマッチング・ファイルのファイル名とファイル・サイズが戻りセットに入って戻されることを指定した制限で構成することができる。スコープを浅い(shallow)と指定すると、指定されたフォルダに入っているドキュメントだけが戻され、深い(deep)と指定すると、指定されたフォルダと、そのサブフォルダ(もしあれば)とに入っているマッチング・ドキュメントが戻される。

インデックスは逆テキスト・インデックス(inverted text index)である。つまり、テキストワード別に編成され、それがキーとなっており、フォルダとドキュメント間の階層関係別に編成され、それがキーとなっていない。その結果、ある照会が処理されるとき、サーチエンジンはインデックスをサーチし、スコープとは関係なく、指定された制限に合致するドキュメントを取得する。スコープ・テスト(scope test)を行うときは、サーチエンジンはドキュメントの検索時にドキュメントの各々について、プレフィックス・マッチング(prefix matching)と呼ばれるストリング(文字列)比較を行い、そのようなファイルがあれば、そのどれが指定のスコープ内にあるかを判断する。見つかったドキュメントで、照会で指定されたものに一致するプレフィックスをもつものは「スコープ内(in scope)」にあると言われる。スコープ内にある、これらのマッチング・ファイルのプロパティはそのあと結果セットに入って戻される。

しかるに、ストリング比較、従って、プレフィックス・マッチングは相対的に低速で、高価なプロセスである。プレフィックス・マッチングは長いファイル名と短いファイル名の両方を使用し、ファイル名の大文字と小文字を区別し、1つのストリングが一意的であるが、同じ意味の複数の表現をもつことができる国際的Unicode(ユニコード)ファイル名を使用しているために、さらに複雑化している。さらに、見つかった各ドキュメントごとに、ドキュメントのフォルダのフルパス(full path)がメモリに作成され、そこではスペースは、パスが任意で未知のストリング・サイズであり、一定の上限がないためにヒープから割り振られている。最後に、2つ以上のフォルダが特定のスコープ・セットで指定される

ことがあるために、また見つかったドキュメントがばらばらになっているために、プレフィックス・マッチングは指定されたフォルダごとに、合致するものが見つかるか、あるいはドキュメントがスコープ内ないと判断されるまで、一度に1ドキュメントずつ行う必要がある。このことは、あるセット内のすべての指定フォルダが、最終的にスコープ内ないと判断されたドキュメントを探すためにプレフィックス・マッチングが行われることを意味し、平均的には、スコープ内にあるドキュメントを探すために指定フォルダの半分がテストされてから、合致するものが見つかった。

以上を要約すると、上述した照会解決手法は、取り扱うファイルの数がわずかであるときは有効であるが、プレフィックス・マッチング・プロセスは、相対的に多数のドキュメントがスコープ・テストされるときは大量のリソースを消費する。これと同時に、階層編成データのOFS、NTFSおよびその他のシステムは、数十万の結果を含んでいる大きな結果セットをサポートする設計になっている。OFSとNTFSおよび類似のシステムはネットワーク構成のクライアント・サーバ環境で使用されるのが代表的であるために、照会が上記のように大きくなり、スコープ・テストに費用がかかっているのが普通である。

発明の目的と概要

従って、本発明の一般的目的はサーチエンジンの照会を大幅に改善する方法およびメカニズムを提供することである。

上記目的に関連する目的は、非常に多数のドキュメントを取り扱う照会に対するサーチエンジンの照会プロセスを大幅に改善することである。

本発明の別の目的は、照会を解決するために要する時間量を大幅に低減する、上記種類に属する方法およびメカニズムを提供することである。

上記目的に関連する目的は、上記の特徴を具備し、基準に合致したドキュメントをスコープ・テストするために要する時間量を大幅に低減する方法およびメカニズムを提供することである。

本発明のさらに別の目的は、既存のサーチエンジンおよび階層編成データと共に機能し、拡張可能である、上記種類に属する方法およびメカニズムを提供する

ことである。

以上を要約して説明すると、本発明はファイル・システムにおけるドキュメントやフォルダなどの、階層編成されたデータ構造の照会に応答し、ドキュメントが指定された基準に基づいて検索されるとき、指定されたスコープに合致するドキュメントが戻されるようにした方法およびメカニズムを提供している。この目的のために、各検索ドキュメント(retrieved document)はカレントドキュメント(current document)として選択され、ドキュメント識別子(document identifier)がそのために取得される。カレントドキュメントの親フォルダの識別子が取得される。親フォルダのIDは、ある種のファイル・システムに存在する、ドキュメントIDから親IDへのマッピング・テーブルにアクセスすることによってドキュメントのIDから判断することができる。このようなマッピング・テーブルはサーチエンジンによって生成することも可能である。

親識別子が取得されると、その識別子は、ハッシュ・テーブル(hash table)のように、キー(親フォルダのID)でインデックスされたフラグを含んでいるデータ構造へのキーとして使用される。フラグはカレントドキュメントを含めて、その親フォルダの下にあるドキュメントがスコープ内にあるか、スコープ内にないかを示す値をもっている。カレントドキュメントがスコープ内にあることをフラグが示していれば、そのドキュメントは、一般的にはファイル名としてそのファイルの他のプロパティと一緒に結果セットに入って戻される。カレントドキュメントがスコープ内にないかをフラグが示していれば、そのドキュメントは戻されない。

カレントドキュメントの親フォルダに対応するエントリがデータ構造になかったときは、プレフィックス・マッチングまたはなんらかの代替テストが親フォルダについて行われ、それがスコープ内にあるかどうか判断される。そのあと、親フォルダのスコープ情報が親フォルダ識別子でインデックスされたフラグとしてデータ構造に追加され、その同じフォルダを共有する以後の検索ドキュメントはプレフィックス・マッチングを受けないで済むようにする。この方法およびメカニズムは各ドキュメントが検索されるつど繰り返され、すべてのドキュメントが上記のようにスコープ・テストされるまで続けられる。

本発明のその他の目的および利点は、添付図面を参照して以下に詳述する説明から理解される通りである。

図面の簡単な説明

図1は、本発明を組み込むことができるコンピュータ・ネットワークを示すブロック図である。

図2は、サーバ・コンピュータとクライアント・コンピュータを含むコンピュータ・システム内のハードウェア・コンポーネントを示すブロック図である。

図3は、サーバ・コンピュータ内のハードウェア・コンポーネントとソフトウェア・コンポーネントを示すブロック図である。

図4は、ファイル・システムまたはサーチエンジンによって維持されるインデックスの一般構造を示す概略図である。

図5は、照会の中で記述された特定の制限に合致するファイルの代表的なリスト例を示す図である。

図6は、フォルダおよびドキュメントの階層ディレクトリ構造の例を示す概略図である。

図7は、ファイル・システムまたは類似システムにおけるドキュメントおよびフォルダの識別子間の階層関係のマッピングを示す図である。

図8は、改良スコープ・テストを容易化するために本発明に従って構築されたキャッシュ・テーブルの例を示すブロック図である。

図9乃至図11は、本発明に従って照会を解決するためにスコープ・テストを行うときとられる一般的ステップを示すフロー図である。

図12は、結果セットに入って戻されるファイル情報の代表的なリスト例を示す図である。

図13は、本発明に従って照会を解決するためにスコープ・テストを行うときとられる代替ステップを示すフロー図である。

図14は、フォルダおよびドキュメントの代替階層ディレクトリ構造の例を示す概略図である。

図15は、改良スコープ・テストを容易化するために本発明に従って構築され

た代替キャッシュ・テーブルの例を示すブロック図である。

好適実施例の詳細な説明

以下、添付図面を参照して説明する。まず、図1を参照して説明すると、図1は本発明を組み込むことができるコンピュータ・ネットワーキング・システム全体を20で示したものである。図示のネットワークはサーバ22と、公知のようにサーバに接続された、コンピュータをベースとする複数のリソース（例えば、パーソナル・コンピュータなどのクライアント・ワークステーション）24₁ - 24_nとを含んでいる。もちろん、以上から理解されるように、本発明はどの特定のネットワーキング環境にも限定されない。つまり、いやしくもネットワークに限定されるのではなく、スタンドアロン型パーソナル・コンピュータなどに組み込むことも可能である。事実、サーバ20の代表例として、相対的にハイパワーのパーソナル・コンピュータがあり、本発明は、どのクライアントからも独立させてサーバ22側のオペレーティング・システム、サーチエンジンおよび/またはファイル・システムに置いておくことが可能である。従って、ここに例示されているサーバ22は独自のC：ドライブ26をもつほかに、D：ドライブ28やE：ドライブ30などの、クライアントがアクセス可能なドライブを物理的にもっているか、あるいはそこに密に接続されている。

コンピュータ・ベースのリソース24₁ - 24_nは相互接続メカニズム32上にネットワーキング方式で相互接続されているが、この相互接続メカニズムはローカルエリアネットワーク、広域ネットワークまたは他の相互接続メカニズムにすることができる。相互接続メカニズム32によると、リソース24₁ - 24_nはサーバ22のドライブD：とE：（28、30）に論理的にアクセスすることができる。物理的には、相互接続メカニズム32はデバイスの各々内のハードウェア・インタフェースに関する選定標準に準拠していれば、どの周知データ伝送媒体でも含むことが可能であり、例としては、同軸ケーブル、電話ケーブル、光ファイバ・ケーブルなどがある。ARCnet（登録商標）、Ethernet（登録商標）およびToken Ring（登録商標）を含む、かかる標準およびその関連ハードウェア・インタフェースはこの分野では周知であるので、ここで詳しく説明することは省

略する。ここでは、多くの商用化インタフェース・ボードはネットワーク方式でコンピュータ・ベースのリソースを接続するために利用できる、とだけ述べておく。

図2に分かりやすく示すように、サーバは、オペレーティング・システム38がそこにロードされているメモリ36に接続されたプロセッサ34を搭載している。オペレーティング・システム38は、Microsoft社のWindows NTTMオペレーティング・システムなどの分散オペレーティング・システムであることが好ましい。本発明によれば、サーバ22は、階層的に編成されたデータ(階層編成データという)、つまり、オペレーティング・システム38と関連づけられているか、またはそこに組み込まれているファイル・システム40内で維持されているような、ドキュメントの構造をそこにもっている。好ましいファイル・システム40は、ドキュメント(つまり、ファイル、オブジェクトなど)がフォルダ(つまり、ディレクトリ、カタログなど)の下に編成できるようなタイプであり、ここでは各ドキュメントはフォルダおよびサブフォルダの階層構成の下に置くことができるが、直属親フォルダは1つだけになっている。サーバ22は、サーバ22を1つまたは2つ以上のネットワーク内デバイスに、キーボードおよび/またはマウスなどの1つまたは2つ以上の入力デバイス44に、およびモニタおよび/またはスピーカなどの1つまたは2つ以上の出力デバイス46に接続するための入出力(I/O)回路42も備えている。サーバ22は図1を参照して上述したC:ドライブ26、D:ドライブ28およびE:ドライブ30などの不揮発性ストレージ48も備えている。

図2にも示すように、ここでクライアント・ワークステーション24₁として示されているリソースの1つまたは2つ以上は、パーソナル・コンピュータなどに行うことができるが、オペレーティング・システム52がそこにロードされているメモリ50を搭載している。なお、オペレーティング・システムは、同様にサーバ22側に常駐するものと同タイプの分散オペレーティング・システムにすることができる。パーソナル・コンピュータ・ベースのクライアント・ワークステーションでは普通であるように、クライアント24₁はプロセッサ53を搭載し、このプロセッサはメモリ50に接続されると共に、クライアント24₁をサ

サーバ22に、キーボードおよび／またはマウスなどの1つまたは2つ以上の入力デバイス56に、およびモニタおよび／またはスピーカなどの1つまたは2つ以上の出力デバイス58に接続するための入出力(I/O)回路54に接続されている。クライアント24₁は独自のローカルC:ドライブ62₁(図1)などの不揮発性ストレージ60(図2)も備えている。

図1と図3に分かりやすく示すように、サーバ22は照会エンジン64を備えており、この照会エンジンを使用すると、ユーザはファイル・システム40(または他の階層編成データ)に照会し、ユーザ入力の照会で指定したものに合致するドキュメントを見つけることができる。照会エンジン64はWindows Explorer(エクスプローラ)または他の類似インタフェースなどの、プログラムを通してユーザとのインタフェースとなることができる。図1に詳しく示すように、クライアント・デバイス24₁〜24_nは各々が独自の階層編成データ(例えば、独自のファイル・システム)またはサーバ22側の階層編成データ(例えば、ファイル・システム40)に照会するための類似の照会エンジン66₁〜66_nを備えている。もちろん、照会エンジンを複製するのではなく、クライアント・デバイス24₁〜24_nに適当なインタフェースを装備させると、そのインタフェースを通してサーバ22の照会エンジン64を利用することも可能である。

図3に詳細を示すように、照会エンジン64は2つのコンポーネントを含んでいる。制限に合致するドキュメントを見つけるための基準マッチング・コンポーネントと、制限に合致するドキュメントの範囲を特定のスコープ内にあるドキュメントに狭めるスコープ・テスト・コンポーネントである。照会エンジン64はファイル・システム40と関連づけられ、ドライブ26、28および30の各々のインデックス72、74および76に影響を及ぼす。同図に示すように、インデックスはある特定ボリュームに対応づけられているが、インデックスは複数のボリュームにスパンすることも可能である。それにもかかわらず、以下で説明するように、本発明は、検索ドキュメントの情報がインデックスにも保存されているかどうかに関係なく、どの検索ドキュメントでも扱うことが可能である。

好ましいインデックス72〜76(図3)は図4に示す編成になっているのが一般で、スペースを節約するためにビット圧縮されたビットのストリームとして

ストアされている。好ましいビット圧縮は本発明の被承継人に承継された米国特許出願第07/986,754号に記載されている。図4に示すように、D:ドライブ・インデックス74のような、各インデックスはテキストワード(KEY₁—KEY_n)をキーとする逆テキスト・インデックスであり、各キー(ワード)はドキュメントのコンテンツ内のドキュメント・プロパティ値またはテキストのどちらかになっている。ドキュメント・プロパティとしては、ファイル・サイズ、作成者、作成日、修正日などのアイテム(項目)がある。

各キーの下位に、識別子、つまり、IDのリストがあり、その各々は特定ボリューム上のドキュメントを整数で一意的に識別している。各IDの下位には、オカレンス(occurrence)、つまり、オフセット・デルタ(offset delta)のリストがあり、これはそのキーがどのドキュメント内のどこに現れたかを示している。従って、図4に示すように、各キーはその後に1つまたは2つ以上のドキュメントIDが続き、各IDはその後にそのIDで示されたドキュメント内のそのキーの1つまたは2つ以上のオカレンスが続いている。類似の逆テキスト・インデックス記憶手法はGerard Salton著「自動テキスト処理—コンピュータによる情報の変換分析と検索(Automatic Text Processing—The Transformation Analysis and Retrieval of Information By Computer)」, Addison-Wesley(1989)に説明されている。

いずれにしても、基準マッチング・コンポーネント68のオペレーションは周知であるので、ここで詳しく説明することは省略する。ここでは、照会エンジンの基準マッチング・コンポーネント68は該当のインデックスなどにアクセスして、ユーザの制限セットに記述された基準を具備するすべてのドキュメントを見つけている、と述べるだけにとどめておく。基準マッチング・コンポーネント68は各ドキュメントの識別子(ドキュメントIDまたはDoc ID)と一緒にドキュメント名を戻し、戻りセットの中で要求されていれば、各ドキュメントの他のプロパティへの直接または間接参照なども戻してくる。図5はこれらのドキュメントのリスト78を示しているが、図には単純化のために人間が読める形式で示され、他のドキュメント・プロパティは示していない。

上記から理解されるように、基準マッチング・コンポーネント68は、指定さ

れたどのスコープにも関係なくインデックス72-76について操作する(一般的にスコープに示された1つまたは複数のドライブを調べてインデックスのどれをサーチすべきかを判断するのではない)。従って、図5はスコープ内のファイルだけではなく、該当のD:ドライブ・インデックスに置かれているすべてのファイルのリスト78を示している。例えば、制限が「そのファイル名に'DOC」があるすべてのドキュメントを見つける」(ただし、「*」はワイルドカード・サーチ用語である)または「そのコンテンツに'budget」という用語があるすべてのドキュメントを見つける」ことを指定していれば、照会エンジンはD:ドライブ・インデックス74をサーチして(スコープがD:\...であったと想定する)見つけた、かかるすべてのドキュメントのリスト78を取得する。ここで注意すべきことは、検索ドキュメントのリスト78が、ある任意の瞬時にストレージに物理的に存在するとは限らないことである。これは、好ましくは、各ドキュメントはその検索時にスコープ・テストされ、結果セットに追加されるか、あるいは破棄されるからである。各ドキュメントをその検索時にスコープ・テストすると、ユーザは、すべてのドキュメントが完全に検索されてから、スコープ・テストされるのを待たなくても、なんらかのサーチ結果(もしあれば)を得ることができるのが普通である。しかし、最初に2つ以上のドキュメントを得てから、あるいは最初に完全リストを得てから、スコープ・テストを行うようにすることも可能である。どの場合も、図5のリストは指定された基準に一致し、スコープ内にある場合もあれば、スコープ内にない場合もあるファイルの集まりを示す概念図である。

以下では、本発明によるスコープ・テスト・コンポーネント70のオペレーションについて詳しく説明する。一般的に、本発明では、見つけた各ドキュメントをプレフィックス・マッチングするのではなく、本発明のスコープ・テスト・コンポーネント70は、フォルダがほとんど常になんらかの階層構造に編成されているという事実、および各フォルダがその下に論理的に配置された複数のドキュメントをもっているのが代表的であるという事実を利用している。従って、本発明の一側面によれば、以下で詳しく説明するように、あるドキュメントのプレフィックス・マッチングの結果を使用することによって、同じ直属親フォルダの

下にストアされている他のドキュメントのスコープ情報は、これらの他のドキュメントで時間のかかるプレフィックス・マッチング・アルゴリズムを実行しなくても取得できるようになっている。

各フォルダは一意的な識別子に対応しており、この識別子は以前にファイル・システムによって割り当てられ、照会エンジンによって影響を及ぼされたものであるか、あるいはサーチエンジンによって一意的に割り当てられたもののどちらかである。例えば、NTFSはあるボリューム上の各フォルダとドキュメントに一意的な識別子を割り当てているが、この識別子はドキュメントIDと呼ばれることがある（フォルダがドキュメントでない場合でもそう呼ばれている）。なお、サーチエンジン64が複数のボリュームにスパンするサーチでドキュメントIDを使用する場合、サーチエンジン64は複数ボリューム間の一意性（固有性）を保つために必要に応じてドキュメントIDだけを変更する。

テーブル、つまり、キャッシュ86は判明したフォルダのスコープに関する情報をストアするために使用される。最初は、キャッシュ86は空であり、親フォルダのスコープに関する情報が分かるに伴って蓄積されていく。もっと具体的に説明すると、キャッシュ86はある親フォルダがスコープ内にあるか、スコープ内にないかを示す"Inscope"フラグを各親フォルダごとにストアするために使用される。キャッシュ86は、好ましくは、メモリ36（図3）内に配列されたハッシュ・テーブルの構造になっている。もちろん、キャッシュ86が十分に大きくなった場合や、以下で説明するように特定のキャッシュの持続性が望ましい場合は、キャッシュの一部または全部を不揮発性ストレージ（図3）に書き出すことができる。周知のように、T.Cormen、C.LeisersonおよびR.Rivest共著「アルゴリズムの紹介（Introduction to Algorithms）」MIT Press,1990の第12章に記載されているように、ハッシュ・テーブルはデータ構造配列になっており、これは実際にストアされるキーの数に比例するサイズになっている。ハッシュ・テーブルはサーチし、そこに追加し、そこから削除することができ、高速平均速度で動作する。もちろん、他のデータ構造をキャッシュ86として使用することも可能である。

次に、図9乃至図11を参照して本発明について説明する。照会エンジン64

は、まず、ユーザから照会の指定を受信すると開始する。例えば、スコープ・セットは、D:\FOLDER1\FOLDER2,D:\FOLDER87と指定することができ、サーチがどちらのフォルダの場合も深いこと、つまり、D:\FOLDER1\FOLDER2,D:\FOLDER87のサブフォルダも、サーチの対象であることを示すことができる。ここで注意すべきことは、NTFSの分散ファイル・システムでは、スコープを可能な限りのドライバとフォルダすべてにわたってグローバルにできることである。

上記に対する応答として、照会エンジン64は、まず、いくつかの判明したフォルダに関するスコープ情報をキャッシュ86に事前に入れておく。つまり、初期値として入れておく(priming)。この例では、フォルダD:\FOLDER1\FOLDER2およびD:\FOLDER87のすぐ下にある検索ドキュメントは、D:\FOLDER1\FOLDER2およびD:\FOLDER87のサブフォルダ情報がまだ分かっていなくても、スコープ内にあることが分かっている。なお、どの任意のフォルダ名からでも、そのドキュメントIDは、ファイル・システム40から、または照会エンジンを通して即時に取得可能である。従って、図6に示すように、D:\FOLDER1\FOLDER2のドキュメントIDは8に等しく、D:\FOLDER87のドキュメントIDは50に等しくなっている。

キャッシュにこの判明情報を初期値として入れるために、ステップ100で、照会解決プロセスは、最初の指定フォルダD:\FOLDER1\FOLDER2のドキュメントIDを取得するが、これは図6に示すように"8"に等しくなっている。ステップ102で、一意的ドキュメントIDである"8"がキャッシュ86(図8)に追加され、Inscopeフラグは真(true)にセットされている。初期値を入れることは、指定フォルダのスコープ情報が初期値としてキャッシュに入るまで、ステップ104の判断に従って続けられる。従って、ステップ106でドキュメントIDの50がD:\FOLDER87について得られ、ステップ102で、キャッシュ86は50をキーとして使用し、Inscopeフラグを真にして初期値が入力される(図8)。

キャッシュに初期値が入力されると、ステップ108で、照会エンジンはマッチング・ドキュメントを検索する(ユーザ指定の基準に基づいて)。この例では、

図5のリスト78に示すように、最初の検索ドキュメントはD:\FOLDER1\FOLDER3\DOC4であり、ドキュメントIDは2に等しくなっている。ここで注意すべきことは、図4に示すように、各ドキュメントのドキュメントIDはインデックス74に入れられ、図5に示すように戻されることである。

従って、各ドキュメントのスコープ・テストは図10のステップ200から開始され、見つけるべきドキュメントが残っていないと、ステップ110(図9)で判断されるまで各検索ドキュメントで続けられる。もちろん、公知のように、ステップ200に先立って、リスト78内の各検索ドキュメントのパスを最初にケース(大小文字)正規化(case-normalized)とロケール正規化(locale-normalized)して短いファイル名にしておく、と、スコープ・テストが高速化される。さらに、いずれかのスコープが深いときは、照会解決プロセスはスコープ・セットにリストされたフォルダを最初に合体しておき、そのセットにも指定されている他のフォルダにネストされたサブフォルダである、フォルダが別々にサーチされないようにする。例えば、D:\FOLDER1のスコープが深いと指定され、スコープ・セットがD:\FOLDER1とD:\FOLDER1\FOLDER4の両方を指定していれば、D:\FOLDER1\FOLDER4の下にストアされたドキュメントもD:\FOLDER1の深いサーチの対象となるので、スコープ・セットをD:\FOLDER1に合体することができる。これは、D:\FOLDER1\FOLDER4が深いと指定されているか、浅いと指定されているかに関係なく有効である。

図7に示すように、および上述したように、各ボリューム、例えば、ドライブD:\ごとに(または複数ボリュームにまたがって)、サーチエンジン64および/またはファイル・システム40は密にパックされたドキュメントIDから親ドキュメントIDへのマッピング配列80、82および84を維持している。この例では、配列はマッピング・テーブル82になっており、そこから任意のドキュメントの親フォルダのドキュメントIDは、そのドキュメントのドキュメントIDをそこへの直接アドレッシング・インデックスとして使用して高速に得られるようになっている。従って、この例では、最初に見つかったドキュメントはD:\FOLDER1\FOLDER3\DOC4であり、ドキュメントIDは2に等

しくなっている(図5)。ドキュメントIDから親ドキュメントIDへのマッピング・テーブル

テーブル82(図7)のインデックスとして"2"を使用すると、直属親フォルダ(D:\FOLDER1\FOLDER3)のドキュメントID("3"に等しい)はステップ202で得られる。なお、NTFSなどのファイル・システムは上記マッピング・テーブル82を各ボリュームごとに直接に維持しているが、マッピング・テーブルに保存されている、上記と同じマッピング情報は、ほとんどの階層編成データまたはファイル・システムからでも直接または間接に取得することが可能である。事実、本発明は、どの特定のドキュメントから親へのマッピングにも限定されず、それとは逆に、どの既存または将来の階層編成データまたはファイル・システムにも拡張可能であり、ドキュメントの親などに関する情報をドキュメント情報から取得または推論することが可能である。

本発明によれば、3に等しいドキュメントIDが取得されると、ステップ204でキャッシュ86は、親ドキュメントIDである3をキーとして使用してサーチされる。この例では、キャッシュ86はこのサブフォルダ情報が初期値として入っていないので、キャッシュ・サーチ・プロセスはステップ204(図10)でヌル(null)、-1または他のエラーコードなどの値を戻し、親フォルダ(キーは3に等しい)は対応するエントリをもっていないことを知らせる。従って、エントリがないので、ステップ206で照会解決プロセスはステップ208にブランチする。ステップ208で、スコープがチェックされ、深いと指定されたスコープがあるかどうか判断されるが、これは、以下で詳しく説明するように、浅い照会と深い照会はこの点では異なる意味合いをもつためである。

この例では、少なくとも1つのスコープは深くなっているため、プロセスは図11のステップ300にブランチし、そこでプレフィックス・マッチング・アルゴリズムなどのテストが行われ、親フォルダ(D:\FOLDER1\FOLDER3)がスコープ内にあるかどうか判断される。公知のように、プレフィックス・マッチングはフォルダのフルパス名を、特定のスコープ・セットに指定されたパス名の各々とストリング単位で突き合わせて比較する。図6に示すツリーか

ら明らかであるように、各フォルダとドキュメントのドキュメントIDはフォルダまたはドキュメントの名前の下にかっこで囲んで示されているが、プレフィックス・マッチングはD:\FOLDER1\FOLDER3がD:\FOLDER1\FOLDER2の

要求されたスコープ内にないと判断している。従って、ステップ300で、D:\FOLDER1\FOLDER3のInscopeフラグに偽の値（ゼロが代表例）が割り当てられ、これはスコープ内にない(not-in-scope)ステータスを示しているので、プロセスはステップ302でステップ304にブランチする。ステップ304でテストが行われ、スコープ・セット内に深いと指定されたフォルダのすべてがプレフィックス・テストされたかどうか判断される。この例では、D:\FOLDER87も深いと指定されているので、ステップ306でD:\FOLDER87が選択され、プロセスはステップ300に戻りD:\FOLDER1\FOLDER3をD:\FOLDER87と突き合わせるプレフィックス・マッチングが行われる。これらのフォルダは合致していないので、Inscopeフラグは偽のままであり、ステップ302で再びステップ304にブランチする。しかし、今回は、D:\FOLDER87は最後に指定されたフォルダであるので、ステップ304でステップ308にブランチし、そこでキャッシュ（ハッシュ・テーブル）86へのキーは親ドキュメントIDである3にセットされる。このあとに続くステップ310で、キャッシュ（ハッシュ・テーブル）86に入るエントリが、Inscopeフラグ値（例えば、ゼロ）に関連するキー3のために作られる。なお、図8はキャッシュ86を示しているが、当然に理解されるように、この時点では最初の3エントリ（つまり、初期値として入力された8、1および50、1のエントリが最近の3、0のエントリと共に）が存在している。

照会プロセスは図10に戻り、ステップ210へ進み、そこでInscopeフラグのテストが行われ、ドキュメントがスコープ内にあったかどうか判断される。ドキュメントはスコープ内になかったため、プロセスは図9のステップ108に戻る（ステップ212にブランチしてドキュメントを結果セットに追加するのではなく）。

図9のステップ108はインデックスにアクセスし、基準に合致する次のドキュメントを取得する。別のドキュメントD:\FOLDER1\FOLDER3\DOC5がリスト78にあるので、ステップ110でプロセスは図10のステップ200に戻り、そこで10に等しい、このドキュメントのドキュメントIDが取得される。

以前と同じように、ステップ202でマッピング・テーブル82 (図7) が参照され、10のドキュメントのドキュメントIDをインデックスとして図7のマッピング・テーブルを使用して、ステップ202はDOC5の直属親フォルダD:\FOLDER1\FOLDER3のドキュメントIDとして"3"を戻す。しかし、今回はループを通るとき、ステップ204でキャッシュ86をサーチすることにより、キー3のエントリがキャッシュ86に存在することが判断され、そこからキャッシュ86はそのInscopeフラグ値(ゼロに等しい)を戻す。従って、プロセスはステップ210にブランチし、そこでInscopeフラグがテストされ、偽であると判断されると、プロセスは図9のステップ108に戻ることになる。従って、以前のドキュメントDOC4と異なり、親フォルダ(D:\FOLDER1\FOLDER3)のスコープはキャッシュに置かれたInscopeフラグから分かっているので、DOC5ではプレフィックス・マッチングは不要である。以上から理解されるように、時間のかかるプレフィックス・マッチング・アルゴリズムを2回以上実行しなくても、キャッシュ86を使用すると、D:\FOLDER1\FOLDER3をその親フォルダ(ドキュメントIDが3である)としてもつ他のドキュメントのスコープが、スコープ内ないと即時に判断することができる。サーチを大きくすると、非常に多数のドキュメントがこの親フォルダを共有できるので、時間節約は何倍にもなる。

ステップ108で次のドキュメントD:\FOLDER1\FOLDER11\MSOFFICE\SHEET4.XLS (図5) が検索されるので、ステップ110でプロセスはステップ200にブランチする。パス・ストリングを分析すると理解されるように、共通の親フォルダD:\FOLDER1\FOLDER11\MSOFFICEをもつ次の3ドキュメントは指定されたスコープ内にはない。上述

したステップに続いて、本発明の一側面によれば、これらの3つの最初のもの(" \SHEET4.XLS")だけが図11に示すようにプレフィックス・マッチングされる。以前と同じように、最初の親フォルダがプレフィックス・マッチングされると、そのキー（例えば、ドキュメントID、従ってD:\FOLDER1\FOLDER11\MSOFFICEのキーは90に等しい）はそのゼロのInscopeフラグと一緒にキャッシュ86に置かれる。次の2ドキュメントについては、親フォルダ90の

エントリはキャッシュ86にあるので、これらのドキュメントのプレフィックス・マッチングは不要になる。なお、プレフィックス・マッチングによると、キャッシュに深いフォルダを初期値として入れておくことは不要になる。

図5に示す例に戻って説明すると、次のドキュメントは"D:\FOLDER1\FOLDER2\FOLDER4\DOC1"であり、ステップ108で検索される。図5に示すように、このドキュメントはドキュメントIDが4になっている。図7のドキュメントIDから親ドキュメントIDへのマッピング・テーブルから、"5"に等しい直属親フォルダのドキュメントIDがステップ202で取得される。ステップ204で、ドキュメントIDである5はキャッシュ86へのキーとして使用される。この時点では、キャッシュ86にはキー5のエントリがないので、キャッシュからはエントリがないことを示す値が戻される。従って、ステップ206でプロセスはステップ208にブランチし、指定されたスコープが深いので、図11のステップ300に進む。

ステップ300で、プレフィックス・マッチング・アルゴリズムが実行され、D:\FOLDER1\FOLDER2\FOLDER4がスコープ内にあると判断する。

その結果、ドキュメントIDが5であるInscopeフラグにInscopeステータスを示す真の値（1が代表的）が割り当てられる。ドキュメントはスコープ内にあるので、ステップ302はステップ308にブランチする。

本発明によれば、ステップ308で、キャッシュ（ハッシュ・テーブル）86へのキーは親ドキュメントIDである5にセットされ、ステップ310でキャッ

シュ86に入るエントリが、真のInscopeフラグ値（例えば、1）に関連するキー5のために作られる。この時点では、図8に示すキャッシュ86には、5個のエントリが入っていることになる。

照会プロセスは図10に戻り、ステップ210へ進み、そこでInscopeフラグが真であるかどうかを判定するテストが行われる。Inscopeフラグは真であるので、ステップ212でDOC1のプロパティが結果セット90（図12）に追加される。なお、この例では、このエントリの戻りセットに指定されたプロパティはドキュメント名、サイズおよび作成日になっている。もちろん、他のプロパティが指定されていれば、これらも戻されることになる（セキュリティ・テスト

のように、あるドキュメントの情報を戻さないとする他の理由がある場合は除く）。そのあと、プロセスは図9のステップ108に戻り、次のドキュメントがあればそれを検索する。

以上から理解されるように、プロセスは図5にリストされている他のドキュメントについても上述したように続け、図12に示すように結果セットを満たしていく。なお、D:\FOLDER1\FOLDER2\FOLDER4\DOC2がテストされるときは、プレフィックス・マッチングは行われず。これはそのドキュメントIDが9であり（図5）、その親ドキュメントIDが5に等しくなっているためである（図7）。その結果、ステップ206でエントリがキャッシュ86に見つかるので、プロセスはステップ210にブランチする。上述したように、Inscopeフラグはこの5のエントリでは真であるので、ステップ210はステップ212にブランチし、そこでDOC2のプロパティが結果セット90（図12）に追加される。従って、以上から明らかなように、あるドキュメントの親フォルダがプレフィックス・マッチングされたあと、キャッシュ86は、プレフィックス・マッチング・アルゴリズムが最終的に親フォルダがスコープ内にあったと判断したか、スコープ内になかったと判断したかに関係なく、同じ親フォルダを共有する他のドキュメントについて以後のプレフィックス・マッチングを行わなくても済むことになる。

最後に、図9のステップ108-110で、マッチング・ドキュメントはすべ

て検索され、スコープ・テストされていることになる。この時点で、プロセスは終了する。

以下では、照会が浅いときの照会リターン・プロセスのオペレーションについて説明するが、そこでは前例がここでも使用されているが、スコープは浅いものと指定されている。スコープが浅いときは、指定された親フォルダのすぐ下にあるドキュメントだけがスコープ内にある。その結果、スコープ・セットに指定されたフォルダの1つのドキュメントIDに等しいドキュメントIDをもつ親フォルダにマッピングされたドキュメントだけがスコープ内にあることになる。例えば、図6に示す階層編成を使用すると、スコープが浅く、D:\FOLDER1\FOLDER2、D:\FOLDER87と指定されていれば、8または50のどちらかの

ドキュメントIDをもつ親フォルダに直接にマップされた検索ドキュメントだけがスコープ内にある。従って、図6に示すように、DOC3だけがスコープ内にある。ここで注意すべきことは、前述の深い照会とは異なり、DOC1とDOC2が浅い照会ではスコープ内にはないのは、これらがFOLDER4の下にあり、従ってFOLDER2の下には間接的にあるだけであるためである。以下で説明するように、この例では、図9と図10のフロー図に例示されているスコープ・テストは他のすべてのドキュメントを除去し、プレフィックス・マッチングを行わずにDOC3だけを結果セットに追加する。

図9のステップ100-106から始まって、キャッシュ86は以前と同じように初期値が入力され、基準はこの例でも同じであるので、ステップ108-110は最終的に図5のリスト78に示すものと同じドキュメントを検索することになる。ステップ200-206で、各ドキュメントのドキュメントIDは以前と同じように親ドキュメントIDを取得し、キャッシュ86をサーチするために使用される。従って、最初のドキュメントはD:\FOLDER1\FOLDER3\DOC4であり、これはドキュメントIDが2になっている(ステップ200)。図2のドキュメントIDから親ドキュメントIDへのマッピング・テーブル82から、ステップ202は3に等しいDOC4の親ドキュメントIDを戻す。3

をキャッシュへのキーとして使用して、ステップ204は3がキャッシュに存在しないとの値または他のエラーコードを戻すので、ステップ206のキャッシュ・エントリ・テストはステップ208にブランチする。

しかし、前述の例とは異なり、ステップ208でスコープは浅いと判断されるので、プロセスはドキュメントをこれ以上テストする必要はない。これはどの浅いフォルダの場合も、初期値が以前にキャッシュ86に真として入っていたためである。直属親フォルダのドキュメントIDは、事前に初期値として入力されたキャッシュになかったため、スコープ内にないことが分かっている。これは初期値入力により、直接にスコープ内にあるすべてのフォルダのIDが追加されていたためである。

従って、ステップ208は図9のステップ108に戻り、そこで次のドキュメントが検索され、同じようにテストされる。図10のステップをたどっていくと

理解されるように、リスト78(図5)内のドキュメントのどれも、D:\FOLDER1\FOLDER2\DOC3を除きスコープ内にない。D:\FOLDER1\FOLDER2\DOC3が検索される時(ステップ108)、ステップ202で検索されたドキュメントIDは8に等しくなる。その時点で、ステップ204は値8のエントリをキャッシュ86内に見つける。キャッシュ86内のすべての浅いエントリは初期値が真として入っているため、DOC3については、ステップ210はステップ212にブランチし、そこでDOC3のプロパティが結果テーブルに追加される。以上から理解されるように、本発明によれば、キャッシュに初期値を入れることにより、すべてのスコープが浅いと指定されている照会では、プレフィックス・マッチングを行う必要がまったくない。

代替実施例では、スコープが深いと指定されているとき、スコープをテストするためにプレフィックス・マッチングを行うのではなく、プロセスは図13のステップ400へ進み、そこでドキュメントのフォルダのパスがフォルダ・ドキュメントIDのセットに変換され、スコープを判断するために分析される。例えば、D:\FOLDER1\FOLDER3はドキュメントID3,1および0(ルート)のセットに対応している。さらに、この代替実施例では、ある種の情報はド

キュメントの祖父フォルダまたは他の上位レベルの親フォルダから取得し、あるいは推論することができる。具体的には、あるドキュメントの直属親フォルダのエントリがキャッシュ86に見つからなかったが、上位レベルの親が見つかり、スコープ内にあって深ければ、スコープ内にある上位親フォルダの下位にある、そのドキュメントおよびすべての判明ドキュメントもスコープ内にある。もちろん、上位レベルの親がキャッシュ86に入っているが、スコープ内になれば、下位レベルのフォルダは、テストがその下位レベルのフォルダで行われない限り、しかもテストが行われるまでは、まだスコープ内にあるかどうかは分からない。

同様に、あるドキュメントがスコープ内ないと判断されたときは、そのドキュメントの上にある上位レベルのフォルダのどれもが、スコープ内になく、深くないと結論することができる。しかし、ドキュメントがスコープ内にあれば、ドキュメントの祖父フォルダ（および／またはその上のもの）が指定されたスコープ内にあることも、スコープ外にあることもあるので、上位レベルの親に関する

有用な情報は分らない。

図13はドキュメントのパスを使用してスコープを判断し、キャッシュを高速に満たす方法を示している。この代替実施例では、図11を図13で置き換えることができる。この例では、その目的上、図14はE:ドライブ30にストアされたフォルダとドキュメントの階層を示し、そこでは各フォルダとドキュメントのドキュメントIDはフォルダまたはドキュメントの名前の下にかっこで囲んで示されている。さらに、次の例では、改良型キャッシュ98（以下で説明するように、浅いスコープを示すフラグを含んでいる）が図15に示されている。次の例では、基準マッチングからはDOC A, DOC B, DOC G, DOC H, DOC M, DOC KおよびDOC Lがこの順序で戻されることを想定している。さらに、照会は浅いと指定されたパスE:\F21\F22または深いと指定されたパスE:\F1\F4 または"E:\F19"内のマッチング・ドキュメントを探し出す。

従って、図15に示すように、キャッシュ98は最初に22,1,1;4,1,0;および19,1,0が事前に初期値として入れられ（図9のステップ100-106）、そこ

では3エントリの各々の中の最後の値は浅いスコープ(1)または深いスコープ(0)を示すフラグになっている。ここで注意すべきことは、InscopeとShallowフラグは1バイトの異なるビットにできることである。図15には4エントリ以上が示されているが、当然に理解されるように、この時点では最初の3エントリだけがキャッシュ86に存在している。

従って、図9のステップ108はDOC Aを検索するが、これは、ステップ200(図10)でドキュメントIDが11(図14)であると判断されたものである。ドキュメントIDから親ドキュメントIDへのマッピング・テーブルなどを使用して、ステップ202はDOC Bの直属親フォルダ・ドキュメントIDが6(図14のフォルダF6を表す)であると判断する。キャッシュ98がステップ204でサーチされるが、6のエントリがないので、プロセスはステップ208にブランチし、そこで少なくとも1つの指定スコープが深いと判断される。この実施例では、プロセスは次に図13(図11ではなく)のステップ400へ進む。

ステップ400はDOC 11のパス、つまり、6, 3, 1, 0内のすべてのフォルダのドキュメントIDを、該当のドキュメントIDから親ドキュメントIDへのマッピング・テーブルなどを使用して取得する。ステップ400は、ドキュメントIDが6であるフォルダF6はキーとして使用されたが失敗していたので、カレントドキュメントIDをパス内の次に上のもの、つまり、親フォルダのドキュメントIDである3にセットする。次のステップであるステップ402は、カレントドキュメントIDを3としてキャッシュ98をサーチする。3はキャッシュ98にないので、ステップ404はステップ406にブランチし、そこでテストが行われ、カレントドキュメントIDがルート・ディレクトリのIDであるかどうか判断され、そうであれば、親はこれ以上存在しないことになる。この例では、他の親が存在するので、プロセスはステップ408にブランチし、そこでカレントドキュメントIDは(親フォルダF1の)親ドキュメントIDである1にセットされる。次に、キャッシュ98は新ドキュメントIDを1としてサーチされる。

図14からと、図13のステップ402-408から理解されるように、キャッシュ98にはこの特定パスのエントリがないので、ルート・ディレクトリがステップ406で見つけられる。この時点で、パス全体がスコープ内にないことが分かっているのは、キャッシュ98にフォルダのドキュメントIDと真のInscopeフラグが初期値として入っていたためである。従って、ステップ410は、それぞれドキュメントIDのキー6、2、1および0に対応するフォルダF6、F2、F1およびルートのInscopeフラグを偽(ゼロ)にしてキャッシュ98に入れることになる。プロセスは図10のステップ210に戻り、そこで偽のInscopeフラグは、プロセスを図9のステップ108に戻し、次のドキュメントを検索させる。

検索される次のドキュメントは図14に示すようにDOC Bであり、ドキュメントIDは12(図10のステップ200)に、親ドキュメントIDは4(ステップ202)になっている。キャッシュ98は初期値が入っているので、4のエントリがキャッシュ98に存在し、Inscopeフラグが真になっているので、ステップ202はステップ210にブランチし、そこからステップ212にブランチする。プロセスは、次に、ステップ212でDOC Bのプロパティを結果セ

ットに追加し、図9のステップ108に戻る。

この例を続けて説明すると、DOC Gがステップ108で次に検索される。図14に示すように、DOC GはドキュメントIDが16(図10のステップ200)に、親フォルダF8のドキュメントIDが8(ステップ202)になっている。このフォルダにはエントリがキャッシュ98(図15)にないので、プロセスのステップ206はステップ208にブランチし、そこで(少なくとも1つの)指定の深いスコープはプロセスを図13のステップ400に進める。

ステップ400で、DOC GのパスのドキュメントID、つまり、8、4、1および0が取得され、カレントドキュメントIDはパス内の次のフォルダのIDである4にセットされる。しかし、今回は、ステップ402はこのドキュメントIDキー4に対して真のInscopeフラグを戻す(以下で説明するように偽の"Shallow"スコープ・フラグと一緒に。これは、フォルダF4のスコープが浅いと指

定されていなかったことを意味する)。従って、ステップ404はステップ412にブランチし、そこでShallowスコープ・フラグがテストされ、プロセスをステップ414にブランチさせる。ステップ414で、カレントドキュメントID4のすべての判明している子供、この例では、ドキュメントIDが8であるフォルダF6だけがキャッシュ98（図15）に追加され、Inscopeフラグは真に、Shallowフラグは偽になっている。これが可能であるのは、フォルダF4がスコープ内にあったことと、スコープが深いために、フォルダF4の下すべての判明しているフォルダもスコープ内にあるためである。この例では、フォルダF8だけがF4の下にあることが分かっているが、他の例では、フォルダのチェイン全体がスコープ内のフォルダの下にあることが分かっている場合があり、その場合は、チェイン全体が追加されることになる。以上から理解されるように、これによりキャッシュ98が高速に満たされる。そのあと、プロセスは図10のステップ210に戻る。

Inscopeフラグは真であるので、ステップ210はステップ212にブランチし、そこでDOC Gのプロパティが結果セットに追加される。プロセスは図9のステップ108に戻り、そこでDOC Hが次に検索される。キャッシュ98にはDOC Gから得た情報が入っているので、次のドキュメントDOC Hがス

コープ・テストされるときは、ステップ200-210（図10）によってスコープ内にあることが即時に判断される。

しかし、次にDOC Mがテストされるときは、DOC Mはスコープ内にない。図10と図13を図14と一緒にたどっていくと理解されるように、DOC Mの直属親フォルダF27と、DOC Mの上位レベル・フォルダ（F21とF23-F26）のどちらも、ドキュメントIDが0であるルート・ディレクトリを除き、この時点ではキャッシュ98にエントリをもっていない。従って、ステップ400-408は、最終的にはステップ404を通してステップ412にブランチすることになる。ルート・ディレクトリは浅いと指定されていないので、判明している子供、つまり、27、26、25、24、23および21には、カレントドキュメントIDのInscopeフラグ値、つまり、偽に等しいルート・ディ

レクトリのInscopeフラグの値が入ることになる。これが可能であるのは、パス全体E:\F21\F23\F24\F25\F26\F27内の各フォルダが実効的にテストされ、上位レベルのフォルダがスコープ内にあると初期値化されていなかったためである。このようにして、キャッシュはこれらのフォルダの偽値で高速に満たされることになる。

DOC Kが次に選択されるときは、キー21（ドキュメントID 21）はすでにキャッシュに偽としてストアされている。その結果、ステップ200-210はプロセスを図9のステップ108に戻し、次のドキュメントを選択させる。

最後に、DOC Lでは、F31はそのためのエントリをキャッシュにもっていない。チェーンのさらに上のF21は偽のエントリをもっているが、このことは必ずしもF31またはF22のどちらかが偽であることを意味しない。従って、プロセスはなんらかの方法でプロセスをショートカットするのではなく、チェーンを上をたどっていく。従って、ステップ200-208はF31がキャッシュ98にないと判断するので、図13のステップ400にブランチする。

ステップ400-404で、ドキュメントIDが22（図14）であるE:\F21\F22はスコープ内にあると判断されるが、そのShallowフラグはそれが浅いと示している。その結果、ステップ412はある種の情報に基づいてステップ416にブランチする。具体的には、正しい合体が行われたとすると、浅いフ

ォルダの上のどのフォルダも深いと指定されていない。そうでなければ、浅いフォルダはもっと深いフォルダに合体されていることになる。さらに、浅いフォルダの下にあるが、検索ドキュメントの上にあるフォルダがスコープ内にあって、深いと指定されていれば、キャッシュ98に初期値を入れておくと、そのようなフォルダは、チェーンを上をたどっていく間に浅いフォルダに到達する前にキャッシュ98に見つかることが保証される。そのようなフォルダは見つからなかったため、現パス内の判明している子供はスコープ内になく、Inscopeフラグを偽にしてキャッシュ98に追加することができる。この例では、F31だけが分かっているので、ステップ416は、Inscopeフラグを0に、Shallowフラグを0に

して31をキャッシュ98に追加する。このようにした1つの目的は、Shallow
スコープ・フラグをキャッシュ98に残しておくためである。なお、正しい合体
が行われていなければ、パスのさらに上の浅いフォルダはステップ404でキャ
ッシュになかったが、ステップ410または414でそのInscopeフラグが偽で
オーバーライトされたことがなかったものとして扱う必要がある。

ステップ416は図10のステップ210に戻り、そこで偽のInscopeフラグ
が図9のステップ108に戻される。しかし、今回は、検索されるドキュメント
は残っていないので、ステップ110を通してプロセスは終了する。以上を要約
すると、図9、10および13の代替実施例では、プレフィックス・マッチング
を行うことなく、正しくスコープ内にあったドキュメントDOC B、DOC G
およびDOC Hだけが戻されている。これと同時に、キャッシュ98は高速に
満たされたので、他のドキュメントを即時にスコープ・テストすることができる
。

以上から理解されるように、上述したどちらの実施例の場合も、プレフィックス
・マッチングが大幅に減少または除去されると共に、キャッシュの埋め込みが
高速化されるので、時間が大幅に節減されることになる。確かに、キャッシュを
使用すると、実際に総時間がスコープ・テストに加わるというシナリオを考える
ことも可能であるが(例えば、フォルダ当たりのドキュメントが1つだけで、そ
の各々を別々にプレフィックス・マッチングするか、長いチェーンを検査する必
要がある場合)、実際にはそのような事態はたとえ起こっても、めったに起こる
ものではない。非常に多数のドキュメントが見つかったときは、本発明によるス

コープ・テストはほとんど常に大幅に高速化する。しかし、このようなまれな事
態のもとでも、本発明を使用したときの時間コストは代表的な照会とフォルダ構
成で本発明を使用すると得られる利点と比べれば、微々たるものである。もちろ
ん、テスト(例えば、見つかったドキュメント数のしきい値をテストすること)
を予備的に行って、本発明に従ってキャッシュ86などを使用すべきか、あるい
はプレフィックス・マッチングを行うべきかを判断するようにした代替メカニズ
ムを考えることも可能である。

同様に、スコープ情報をインデックスに含めて、基準マッチングとスコープ・テストを実効的に1つのプロセスに結合することも可能である。しかし、スコープ情報をインデックスに含めると、これに関連してコスト上の問題や他の問題が起こることになる。例えば、これを行う1つの方法は、インデックス内の各フォルダに個別的に名前をつけることである。この場合、1つのキーが各フォルダ名になる。しかし、サーチは多数のサーチ・キーを必要とするため、サーチに大量の時間が加わることになる。別の方法として、各パスを複合キーとしてストアすることも可能である。しかし、例えば、ファイルまたはフォルダが改名されたときは、そのたびに大量の更新をインデックスに対して行い、インデックスを最新状態に保つ必要がある。以上を要約すると、スコープ情報をインデックスに入れることは可能であっても、実用的であるとは考えられない。

最後に、キャッシュ内にまたは結果セット内に入っているフォルダまたはドキュメントは照会が解決されている間に作成、改名または削除することができる。さらに、以上から理解されるように、あるスコープが与えられているとき、キャッシュを不揮発性ストレージに持続的にストアしておくことが可能である。例えば、ユーザがそこから選択できるスコープの数が制限されている場合、利用可能な各スコープごとにキャッシュを持続的にストアしておくことができる。同じように、1つまたは2つ以上の特定スコープがユーザによって頻繁に選択される場合、最も頻繁に選択されるスコープごとに1つまたは2つ以上のキャッシュをストアしておくことができる。従って、あるキャッシュの結果セットが将来の使用に備えて持続的にストアされているような場合には、ドキュメントまたはフォルダのステータスはストアしてから将来使用するまでの間に同様に変化することがある。

上記のいずれの場合も、結果セットまたはキャッシュをフォルダまたはドキュメントのステータス変化に基づいて更新することが可能である。この目的のために、Windows（登録商標）には、照会または他のプロセスに通知を行うことができるアプリケーション・プログラミング・インタフェース（API）が用意されている。これらのWIN32-APIは、NTFSではFindFirstChangeNotificationおよ

びFindNextChangeNotificationと名づけられており、なんらかの変更がフォルダまたはドキュメントに行われたとき、変更されたフォルダまたはドキュメントのドキュメントIDと変更のタイプ（例えば、改名や削除）を含む変更通知を行う。

これらのAPIを使用すると、照会プロセスはドキュメントへの変更をモニタすることができる。例えば、削除通知を受け取ると、対応するドキュメントIDを探すためにハッシュ・テーブル・キャッシュがサーチされ、エントリが見つかり、そのエントリのスコープ情報は有効でなくなったのでエントリが削除される。改名通知を受け取ったときは、ハッシュ・テーブルが同じようにサーチされる。ドキュメントIDに対応するエントリがハッシュ・テーブルに見つかったときは、その改名アイテムはフォルダでなければならないので、プレフィックス・マッチングが新しいフォルダ名で行われ、そのフォルダのInscopeフラグを更新できるようにする。最後に、以上の説明から理解されるように、結果セットは、たとえ照会がまだ解決中であっても、同じように調整可能であり、照会が解決中である場合には、まだスコープ・テストされていない検索ドキュメントは上記APIを通して同じように、追加、削除および／または改名することができる。

上述してきた詳細説明から理解されるように、本発明が提供する方法およびメカニズムによると、特に非常に多数のドキュメントを取り扱う照会では、ファイル・システムの照会が大幅に改善される。本発明の方法およびメカニズムによれば、照会を解決するために要する時間量が大幅に低減され、基準合致ドキュメントをスコープ・テストするために要する時間量が大幅に低減される。本発明の方法およびメカニズムは既存のファイル・システムと一緒に機能し、拡張が可能である。

本発明は種々態様の変更および代替構成が可能であるが、本発明の実施例のいくつかを図示し、詳しく上述してきた。しかし、当然に理解されるように、本発明は上述してきた具体的な実施態様に限定されるものではなく、本発明の精神と範囲に属するあらゆる変更または改良、代替構成、および等価技術を包含するものである。

【図1】

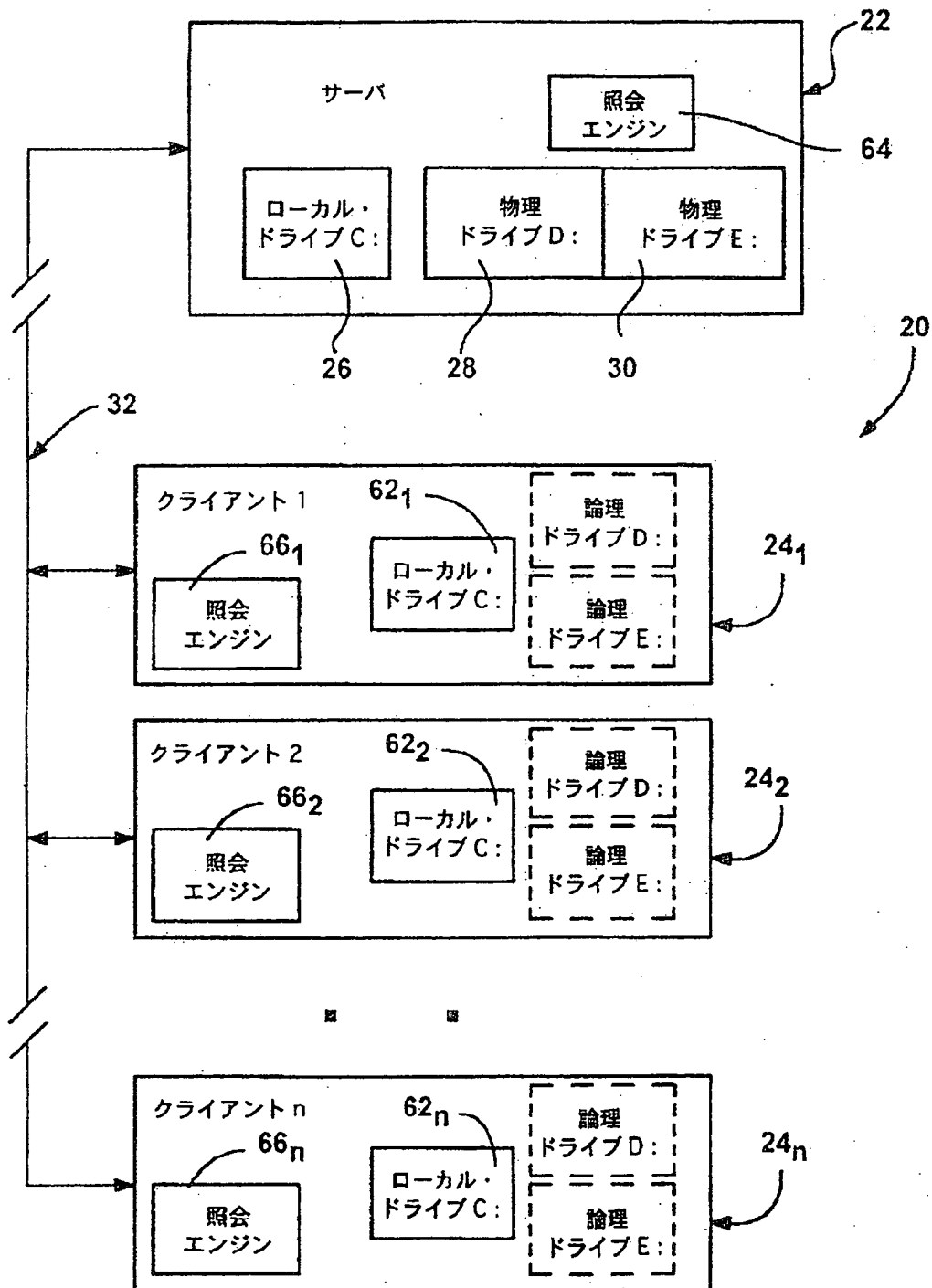


FIG. 1

【図2】

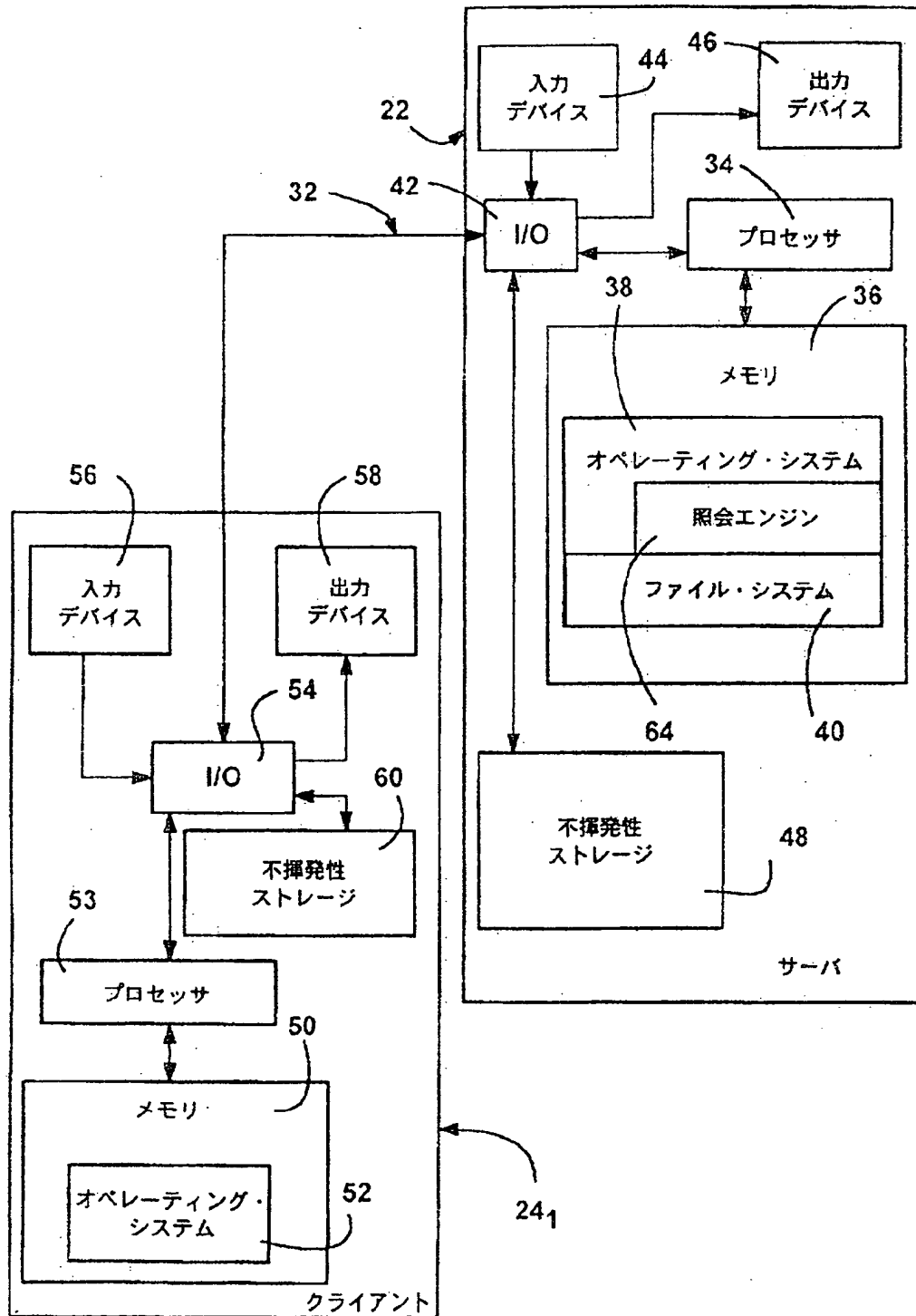


FIG. 2

【図3】

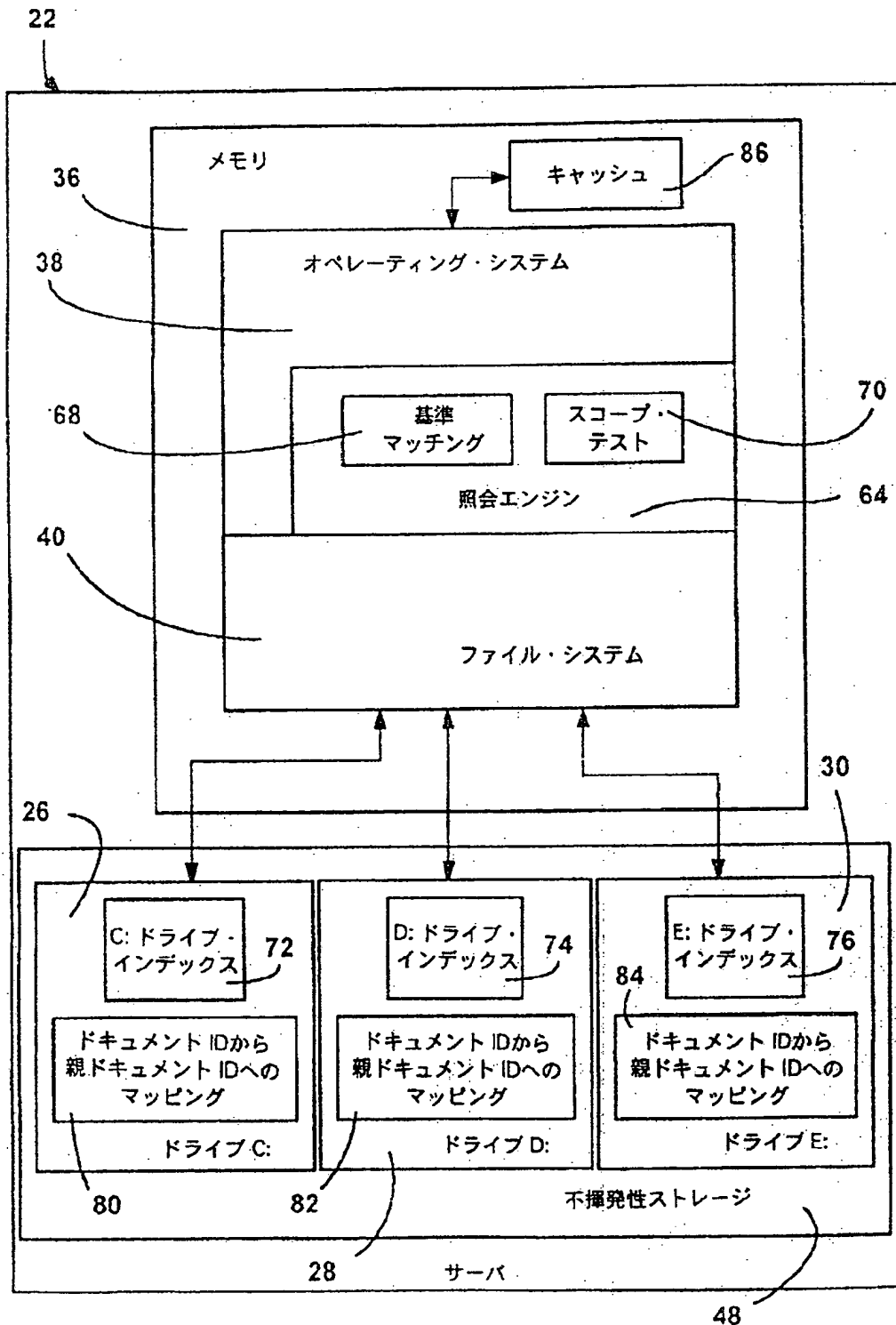
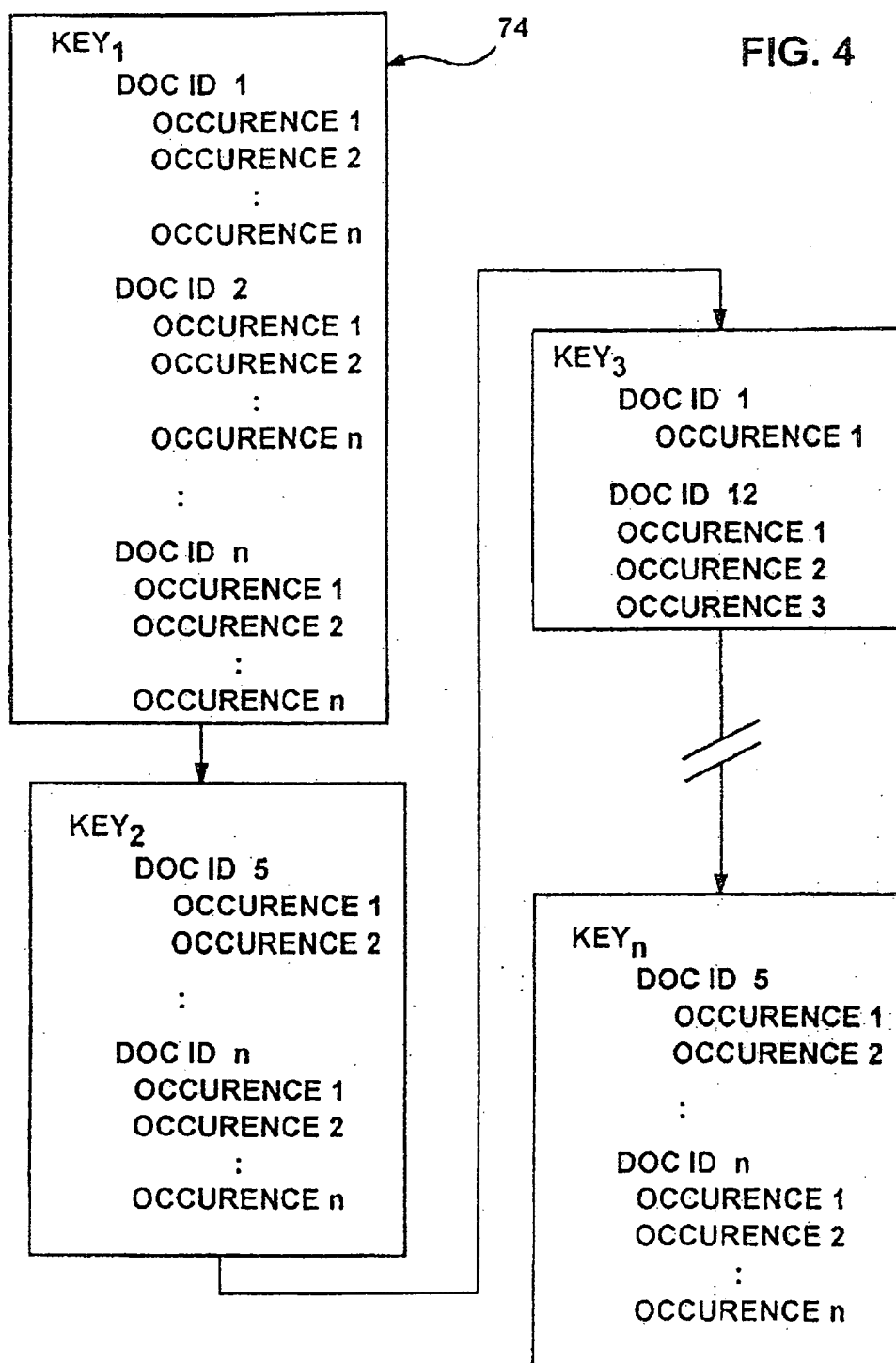


FIG. 3

【図4】



ドキュメント名	ドキュメントID
D:\FOLDER1\FOLDER3\DOC4	2
D:\FOLDER1\FOLDER3\DOC5	10
D:\FOLDER1\FOLDER11\MSOFFICE\ISHEET4.XLS	119
D:\FOLDER1\FOLDER11\MSOFFICE\WORDPROC.DOC	137
D:\FOLDER1\FOLDER11\MSOFFICE\ISHEET8.XLS	159
D:\FOLDER1\FOLDER2\FOLDER4\DOC1	4
D:\FOLDER1\FOLDER7\MYDOCS\ABC.TXT	145
D:\FOLDER1\FOLDER8\DOCABC	55
D:\FOLDER1\FOLDER7\DOC27.DOC	81
D:\FOLDER1\FOLDER31\FOLDER27\FOLDER90\FOLDER18\DOC8	337
D:\FOLDER1\FOLDER31\FOLDER27\FOLDER90\XYZ.XLS	221
D:\FOLDER1\FOLDER2\FOLDER4\DOC2	9
D:\FOLDER1\FOLDER6\A.DBS	300
D:\FOLDER1\FOLDER5\A123.DOC	67
D:\FOLDER1\FOLDER8\UTX.XLS	45
D:\FOLDER1\FOLDER6\123.TXT	57
D:\FOLDER1\FOLDER6\KLM.TXT	48
D:\FOLDER1\FOLDER2\DOC3	6
D:\FOLDER1\FOLDER7\SCHEDULE	88
D:\FOLDER1\FOLDER9\LOG.TXT	94

562

【図6】

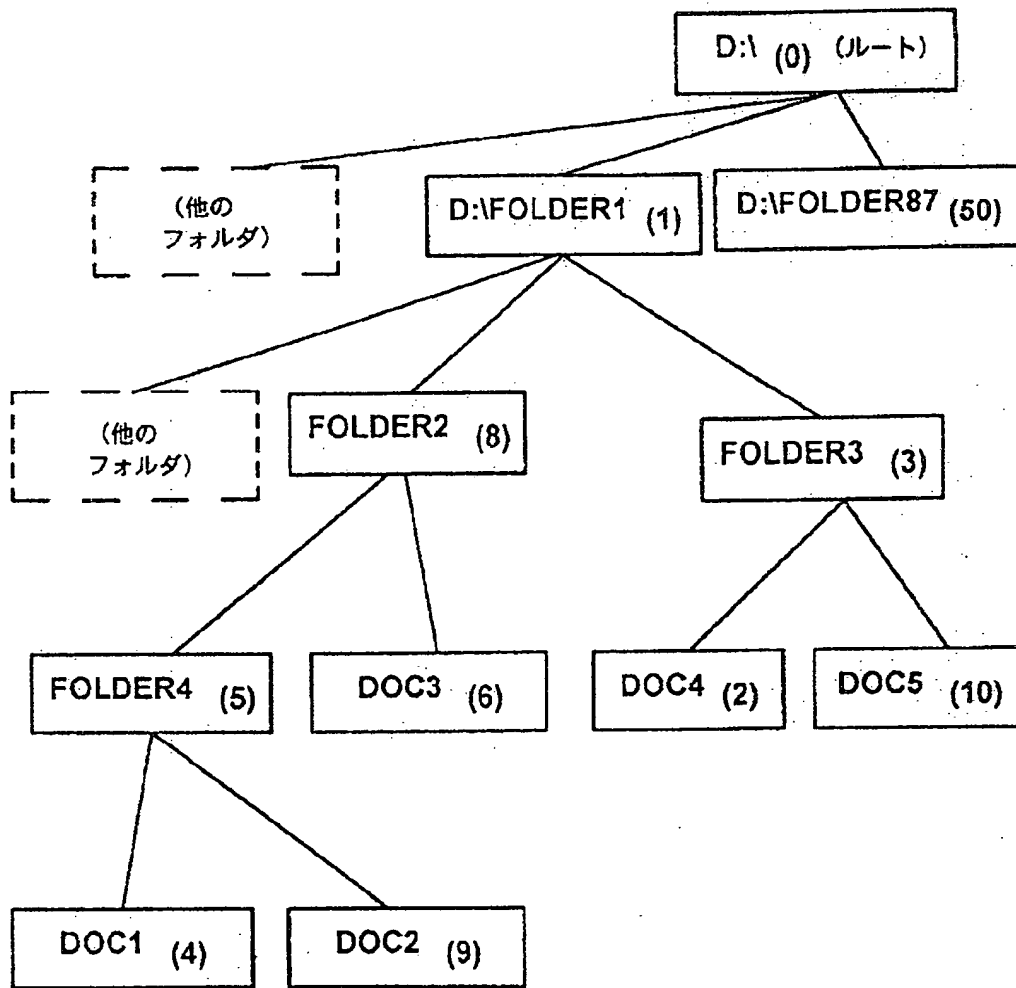


FIG. 6

【図7】

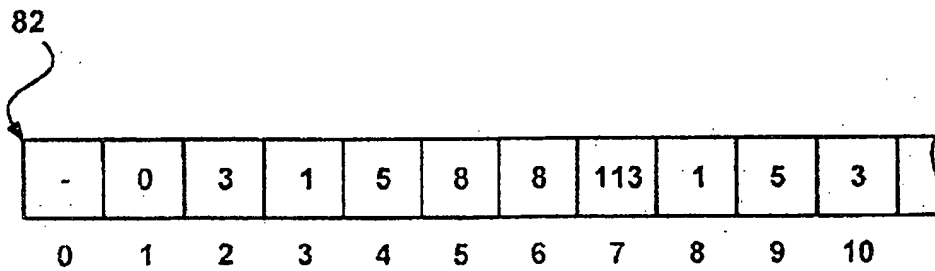


FIG. 7

【図8】

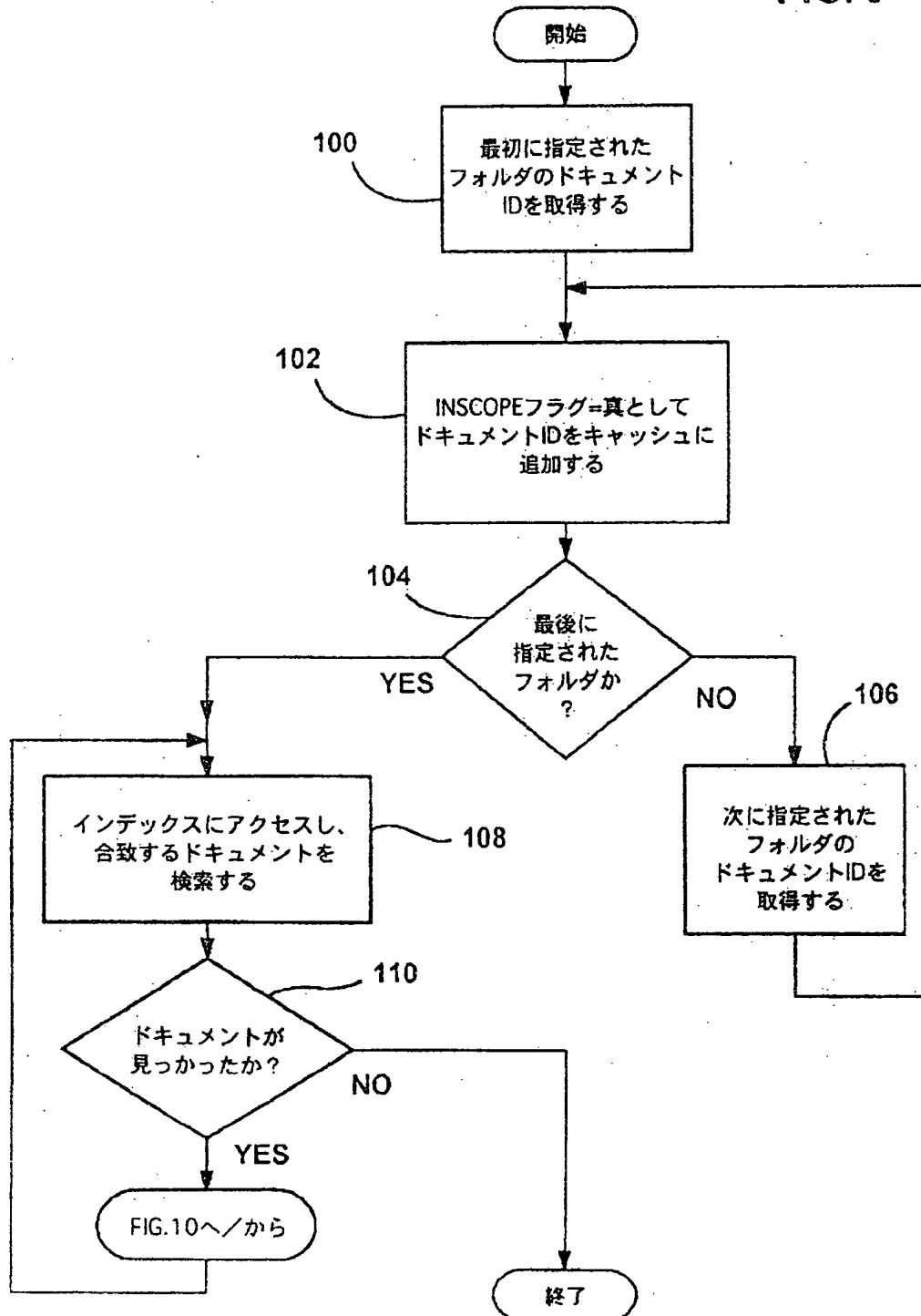
86

8	1
50	1
3	0
90	0
5	1
864	0
193	0
553	0
⋮	⋮
881	0
239	0
363	0
193	0
889	0
765	0
773	0

FIG. 8

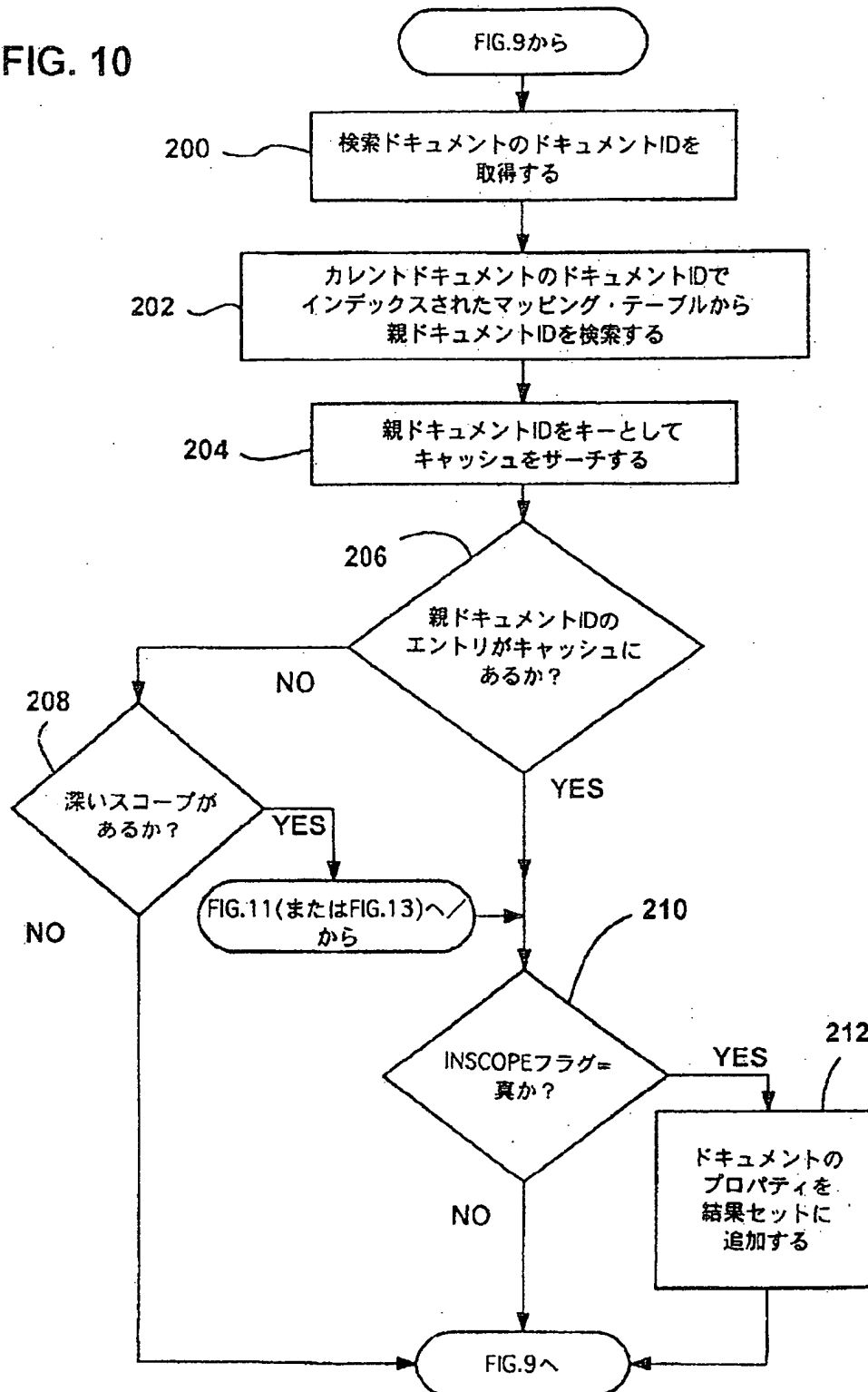
【図9】

FIG. 9



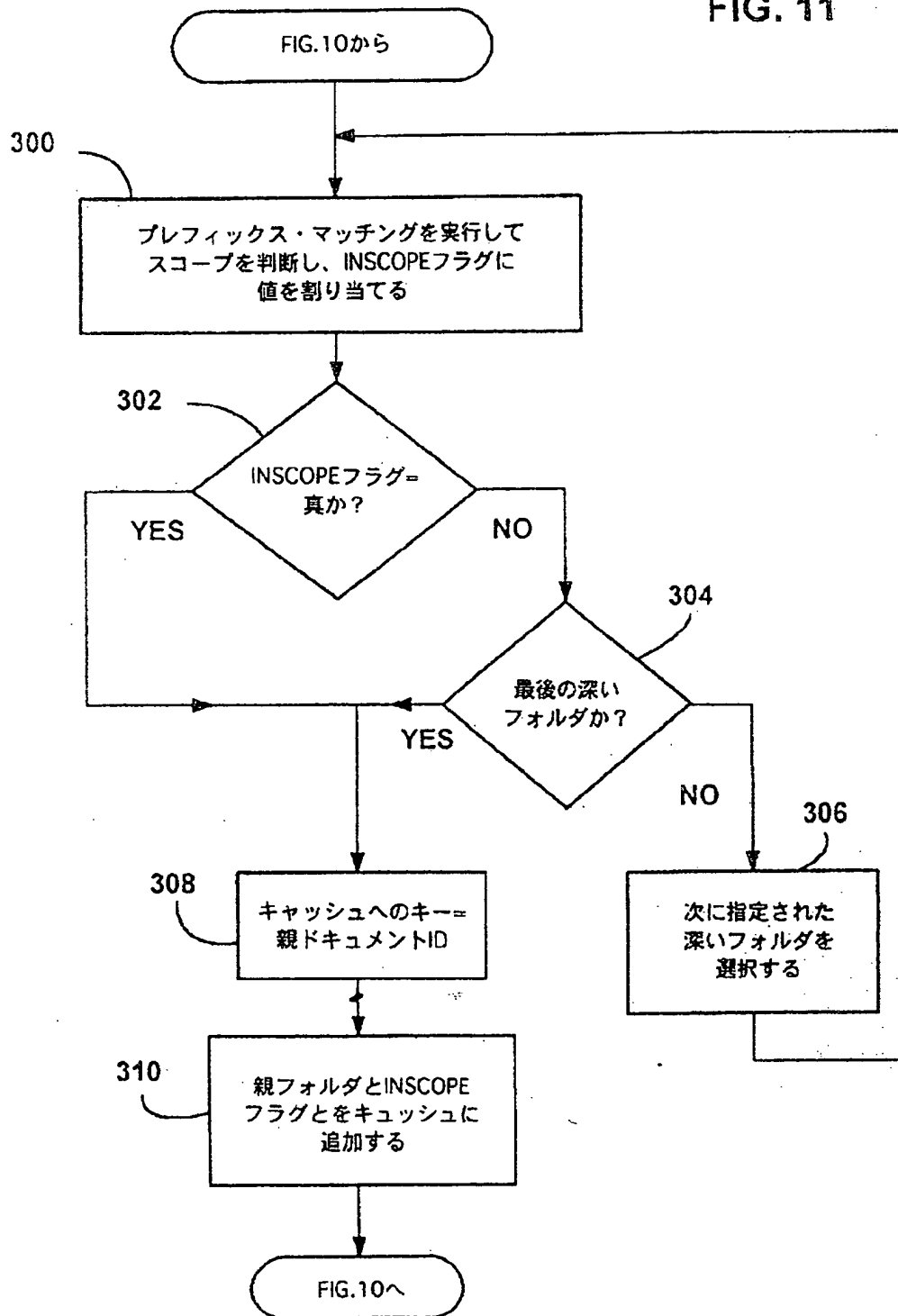
【図10】

FIG. 10



【図11】

FIG. 11



【図12】

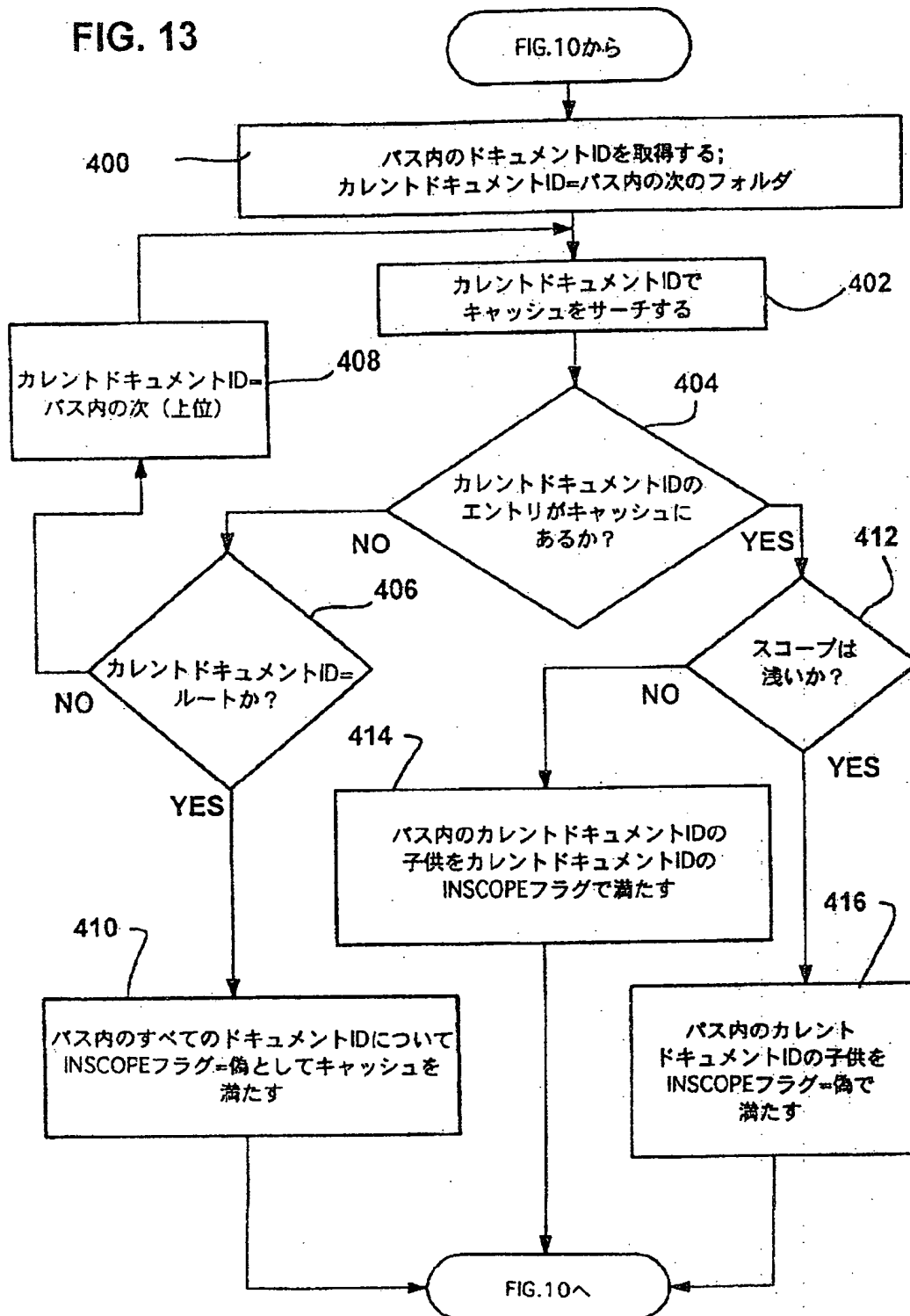
90

<u>ドキュメント名</u>	<u>サイズ</u>	<u>作成 日</u>
D:\FOLDER1\FOLDER2\FOLDER4\DOC1	21,594	04/16/94
D:\FOLDER1\FOLDER2\FOLDER4\DOC2	453,197	07/09/95
D:\FOLDER1\FOLDER2\DOC3	48,591	03/13/96

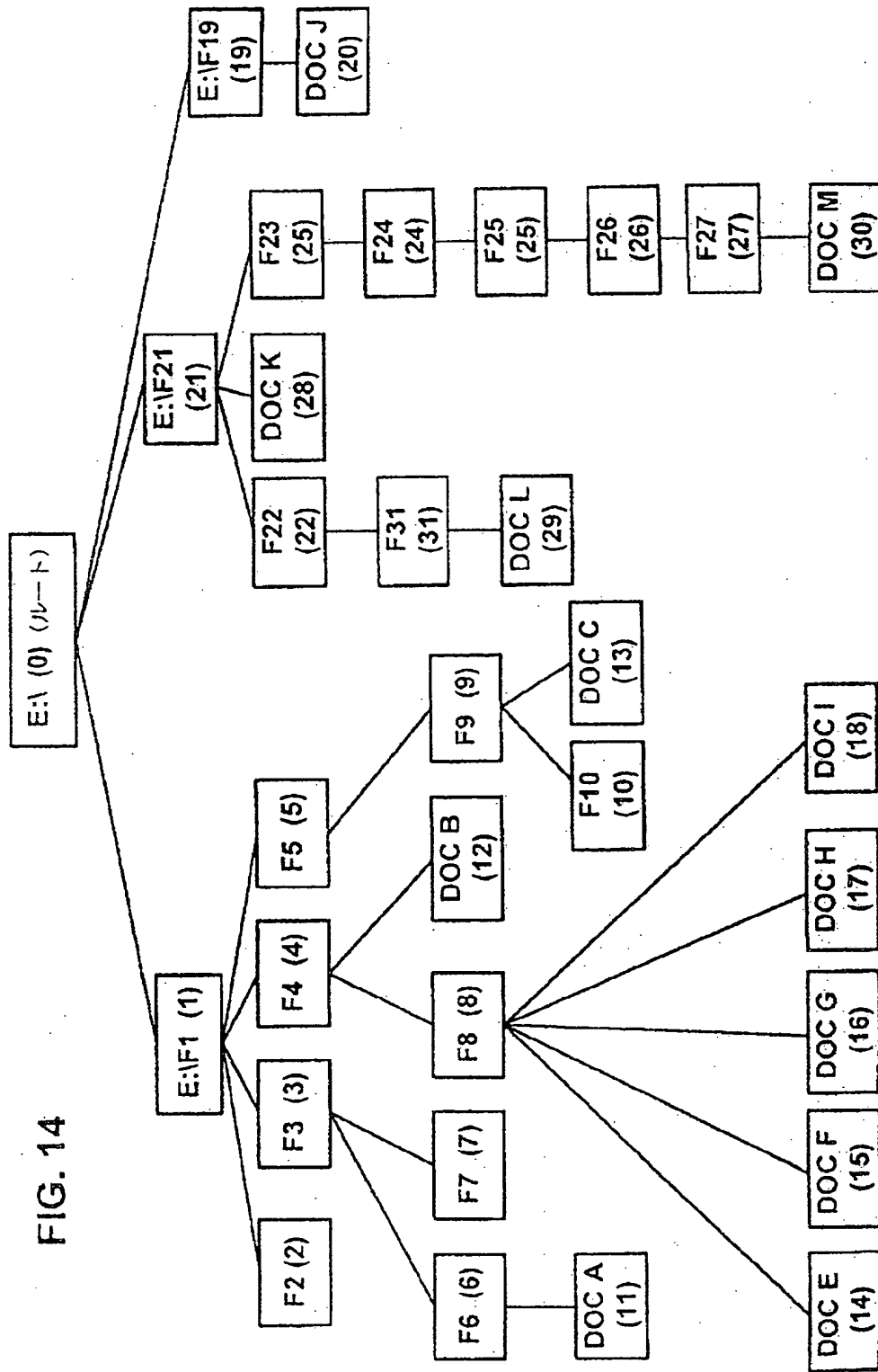
FIG. 12

【図13】

FIG. 13



【図14】



【図15】

98

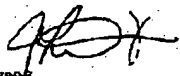
22	1	1
4	1	0
19	1	0
6	0	0
3	0	0
1	0	0
0	0	0
8	1	0
27	0	0
26	0	0
25	0	0
24	0	0
23	0	0
21	0	0
31	0	0

FIG. 15

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/04568

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 17/30, 15/40 US CL : 707/3.4, 100 According to International Patent Classification (IPC) or to both national classification and IPC																				
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 707/100 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS, PROQUEST, ACM, DIALOG																				
C. DOCUMENTS CONSIDERED TO BE RELEVANT																				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.																		
Y	US 5,729,730 A (WLASHIN et al) 17 Mar 1998, cols. 6-11.	1-5, 9, 12, 15, 16, 19 and 21																		
Y	US 5,319,762 A (MAYER) 07 June 1994, col. 17.	1-5, 9, 12, 15, 16, 19 and 21																		
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.																				
<table border="0"> <tr> <td>* Special categories of cited documents:</td> <td>* T</td> <td>these documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention</td> </tr> <tr> <td>* A</td> <td>document defining the general state of the art which is not considered to be of particular relevance</td> <td>* X</td> </tr> <tr> <td>* E</td> <td>earlier document published on or after the international filing date</td> <td>* Y</td> </tr> <tr> <td>* L</td> <td>document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified)</td> <td>* Y</td> </tr> <tr> <td>* O</td> <td>document referring to an oral disclosure, use, exhibition or other means</td> <td>* A</td> </tr> <tr> <td>* T</td> <td>document published prior to the international filing date but later than the priority date claimed</td> <td>* A</td> </tr> </table>			* Special categories of cited documents:	* T	these documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	* A	document defining the general state of the art which is not considered to be of particular relevance	* X	* E	earlier document published on or after the international filing date	* Y	* L	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified)	* Y	* O	document referring to an oral disclosure, use, exhibition or other means	* A	* T	document published prior to the international filing date but later than the priority date claimed	* A
* Special categories of cited documents:	* T	these documents published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention																		
* A	document defining the general state of the art which is not considered to be of particular relevance	* X																		
* E	earlier document published on or after the international filing date	* Y																		
* L	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified)	* Y																		
* O	document referring to an oral disclosure, use, exhibition or other means	* A																		
* T	document published prior to the international filing date but later than the priority date claimed	* A																		
Date of the actual completion of the international search		Date of mailing of the international search report																		
08 JUNE 1998		80 SEP 1998																		
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer MICHAEL WALLACE  Telephone No. (703) 308-8996																		

フロントページの続き

(72)発明者 マイルスキー、バートズ、ビー、
アメリカ合衆国 98125 ワシントン州
シアトル 46ティーエイチ アヴェニュー
ノースイースト 10052

【要約の続き】

ルダで行われ、それがスコープ内にあるかどうか判断
される。そのあと、親フォルダのスコープ情報は親フォ
ルダのドキュメント識別子でインデックスされたフラグ
としてデータ構造に追加される。